パソコンテレビX1シリーズ
X1/C/D/F/turbo
ハードウェア解析
X1
システム研究室
おもしろマシンのブラックボックス探検
有田 隆也
牛嶋 昌和
Itti Rittaporn
日本ソフトバンク

システム研究室

おもしろマシンのブラックボックス探検

有田 隆也
牛嶋 昌和
Itti Rittaporn

先進マシンの未知領域に向けて

数多いパソコンの中でも，X1シリーズほどお茶の間の風景にとけこむマシンはありません。他の多くのパソコンが，スピード，メモリ容量など，第1次元的いわば垂直の競争に明け暮れている中で，X1は横への広がりを，具体的にいえばテレビやビデオなどAV機器を取り込む可能性を示しその登場の印象は鮮烈でした。

つづくX1 turboの出現もまた衝撃的でした。

turboという言葉は，今や掃除機の商品名になるまで 一般的な用語になりました。しかし，X1 turboのそれはturboの名に決して恥じないものであり，マシンの解析を進めるにつれますますその確信を深めました。

X1 turboの機能や性能の先進性については，もはや多くを語る必要はないでしょう。しかし同時に注目されるのは従来のX1シリーズとの互換性を保つため，血のにじむような努力のあとを随所に発見したことです。

この本のねらいは，X1/C/D/F/turboを具体的に解剖することを通じ「パソコンとはいったいなになのか？」を，ソフトウェアやハードウェアにとらわれず考えてみようというところにあります。

ソフトもハードも考え方そのものはまったく同じであり，両者は切り離すことのできないものと考えています。パソコンの本質的・原理的な問題を順序立てて追究していけば，読者の方にもこのことが理解していただけると期待しています。

基本的な部分を重視した構成をとり，「マシン語やハードウエア」の入門の入門書的な性格を持たせました。

初心者の方にも無理なく読んでいただけることでしょう。

1985年 初夏　東京

有田隆也

CONTENTS

Ⅳ　ゆたかな周辺機器と拡張性

I　X1シリーズへの招待

●

コンピュータってなに？
コンピュータの仕組み
コンピュータの動作
先進のハードウェアX1の構成

コンピュータってなに?

■パソコンの実体を知ろう

「コンピュータって，いったいなに？」という質問に対して「しょせんブラックボックスさ」という答えが少なからず返ってくるかも知れません。

こんな答えにも説得力があるほど，最近のパソコンは中身の知識なしに，フロッピディスクやカセットテープのソフトを買ってきて，画面に示される親切な指示に従うだけで十分使いこなせるようになってきています。

パソコンショップで販売されているソフト(とくにX1用のもの)は，このところ非常にレベルアップしています。だから，ソフトを買ってきて楽しむだけなら，パソコンの仕組みや装置のことまでとくに意識する必要もないかも知れません。

しかし，いったんパソコンの素晴しい性能を知れば知るほど，いったいどうしてこんなちっぽけなマシンにこれほどのことができるのかと不思議になってきます。そして，だれでも一度は中身をのぞいてみたくなるでしょう。

この好奇心こそ，パソコンを真にパーソナルなものにする源です。少なくとも少年時代に誰もが持っていたような科学的探究心をもって機械の中身を探検していくと，そこには思いがけない発見が待っています。よくここまで考え工夫したものだ，という人間の知恵と技術に対する感動があなたをとらえることでしょう。

この感動が，いやでもパソコンをより身近なものとするに違いありません。また，パソコンの中身を実感をもって理解できるようになれば，市販のソフトを使う場合の満足感も一段と増してくることに気付くでしょう。

いわばパソコン感覚とでもいうものが身について，できあいのソフトを利用するにせよ，自分でプログラミングするにせよ，より一層の使いこなす喜びを味わうことができることでしょう。

何はともあれ，パソコンを構成している基本的な要素から理解をしていきましょう。むずかしそうに見えるハードウェアの話も，基本的で簡単な話が少し積み重なっただけのことですから……。

■情報はどう伝達されるか

X1のケースのフタをはずしてみると，四角い緑色の基板がむずかしそうな顔をして納まっているのがみえます。その基板の上にはたくさんのチップ(ICやLSI，単に“石”ともいう)が所狭しとならんでいて，その間を無数の細かい線が走っています。

この導線を伝わって信号がすさまじい速度でIC(集積回路)の間をかけ巡るのです。ここで演算・処理される信号は，高い電圧と低い電圧のどちらかという2値信号です。この信

号は高い電圧を「1」，低い電圧を「0」に対応させて表現します。1と0は論理上の真と偽にもそれぞれ対応しています。論理と電圧，つまりソフトウェアとハードウェアはこのように明快に対応しているといえます。

ひとつの導線はある時点においては「0」か「1」のどちらかしか伝えることができません。この最小の情報単位を「1ビット」といいます。

0と1に関する論理演算に時間的な考え方を取り入れ，電子回路素子で表現したものがコンピュータであるともいえます。

ICの間をデータが移動する際，同時に何本かまとめて伝達したほうが効率的です。1本だけだと0と1のふたつの状態しか表せませんが，2本(AとB)あれば，次のように4通りの状態を表すことができます。

A＝0　B＝0

A＝0　B＝1

A＝1　B＝0

A＝1　B＝1

コンピュータの中では，データを8の倍数，たとえば8本とか16本をひとまとめにするのが普通です。このような信号線のひとまとまりを「バス(bus)」といいます。そして，とくに8ビットのデータ(0あるいは1が8個集まったもの)を「1バイト(byte)」と呼びます。

ここでコンピュータ入門では欠くことのできない2進数の知識が必要となってきますが，実際のバスの状態と対応させると理解しやすいでしょう。

そこで信号線の束の1本1本を区別するために0，1，2，3……と番号をふり，導線での状態と2進数を対応させてみます(図I-1)。

図I-1　物理的と論理的状態の対応

このように物理的状態が高電圧のケタを「1」，低電圧のケタを「0」として，番号のとおり0と1を並べたものが2進数です。バスの状態によって伝わるデータは，

$10010110_{(2)}$

ということになります。『(2)』と右下に小さくあるのは，2進数であることを明示したものです。

われわれにとっては10進数があたりまえになっています。これは両手の指の数が10本であるからなどといわれていますが，0と1しかないコンピュータには2進数はきわめて自然な考え方です。

では10進数と2進数を対応させてみましょう。

2進数	10進数	2進数	10進数
0…………	0	101…………	5

1…………1		110…………6	
10…………2		111…………7	
11…………3		1000…………8	
100…………4		1001…………9	

ここでたとえば2進数の11は，バス番号の0と1だけが高電圧でほかのバスは低電圧でであり，上位のケタの0……が省略されていることに注意してください。

一般に $xyz_{(n)}$ と表記されたn進数の値は，

$x \times n^2 + y \times n + z$

となります。たとえば $110_{(2)}$ は，

$1 \times 2^2 + 1 \times 2 + 0 = 6$

となるわけです。

2進数は0と1しか使わないので，みるみるうちにケタ数が増えスペースを取りすぎてしまい，また混乱も招きます。そこで対策として登場するのが16進数です。

2進数では0と1の2個

10進数では0～9の10個

の数字を使用しますが，16進数では当然16個の数字が必要なのです。既成の数字は10個しかありません。そこでアルファベットのAからFの6個を，16進数における10から15に対応させて使うことにしています。

A＝10，B＝11，C＝12，D＝13，E＝14，F＝15

16進数であることを区別するために，一般的に数字の最後にHをつけます。

16進数は次のように表現します。

2F0H

また10進数に直すのは，さきほどの例と同じように，

$2 \times 16^2 + 15 \times 16 + 0 = 752$

とします。

ところで16進数はあくまでも，2進数の表記上の便宜から使われますので，2進数と16進数の対応をしっかりつかんでおくことが大切です。つまり4ケタの2進数が0から始まって16個の数を表すので，2進数の4ケタ分を16進数の1ケタに対応させればよいのです。

2進数	16進数	2進数	16進数
0000…………	0	1000…………	8
0001…………	1	1001…………	9
0010…………	2	1010…………	A
0011…………	3	1011…………	B
0100…………	4	1100…………	C
0101…………	5	1101…………	D
0110…………	6	1110…………	E
0111…………	7	1111…………	F

別にこの対応を覚える必要はありません。たとえば$1100_{(2)}$は10進数では「8＋4＋0＋0＝12」だから12を16進数に直してCとしてもかまいません。

この対応を用いると，次のように2進数を16進数に変換することができます。

$\underbrace{0111}_{7}\underbrace{1110}_{E}{}_{(2)} \longrightarrow$ 7EH

$\underbrace{1100}_{C}\underbrace{0100}_{4}\underbrace{1001}_{9}\underbrace{0111}_{7}{}_{(2)} \longrightarrow$ C497H

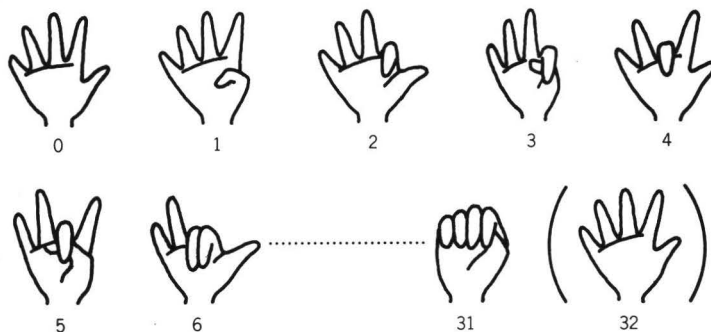
column1

片手と2進数

5本の指を使って，いくつまで数えることができますか。普通は親指から折っていき，次に小指から開いていくというふうにして10まで数えますね。ところが2進数を応用すると32まで数えることができます。すなわち1本の指は伸ばしている状態（これを0とする）と，折っている状態（これを1とする）のふたつがあります。指は5本ですから，2進数で行うと，

$2^5=32$

まで片手で数えられるというわけです。神経回路網にもとづく立体視を研究している野中さんは，実際図のようにして，かなりのスピードで32まで数えます。あなたもぜひチャレンジしてみてください。

野中さんの数え方

人間の脳と2進数

コンピュータは，0と1だけで情報を伝達しますから，いかにも非人間的なクールさを感じます。けれども，人間の脳を形成している神経細胞も，それと同じような特性があることが医学的に明らかにされています。

すなわち神経細胞は，興奮して次の段の神経細胞に1を伝えるか，興奮しないで0の状態のままであるかという，2進数的伝達手段を持っているというのです。そうしたひとつひとつの神経細胞が無数に集まると，並列性と階層性を持つことによって，高度な知的情報処理ができるのだということです。

このあたりを深く考えていくと知的興奮をおぼえると同時にいささか不気味な思いにもとらわれます。

こんどは逆に16進数から2進数へ変換する場合も，4ケタずつ区切って行います。

```
 3    A
┌─┐  ┌─┐
0011 1010   ──→ 00111010(2)

 F    0    2    3
┌─┐  ┌─┐  ┌─┐  ┌─┐
1111 0000 0010 0011   ──→ 1111000000100011(2)
```

参考までに10進数，2進数，16進数の対応表(図 I-2)をあげておきます。

図 I-2　2，10，16進数対応表

10進数	2 進 数	16進数	10進数	2 進 数	16進数
0	00	0	17	10001	11
1	01	1	18	10010	12
2	10	2	19	10011	13
3	11	3	20	10100	14
4	100	4	21	10101	15
5	101	5	22	10110	16
6	110	6	23	10111	17
7	111	7	24	11000	18
8	1000	8	25	11001	19
9	1001	9	26	11010	1A
10	1010	A	27	11011	1B
11	1011	B	28	11100	1C
12	1100	C	29	11101	1D
13	1101	D	30	11110	1E
14	1110	E	31	11111	1F
15	1111	F	32	100000	20
16	10000	10	33	100001	21

コンピュータの仕組み

■CPU—中央演算処理装置

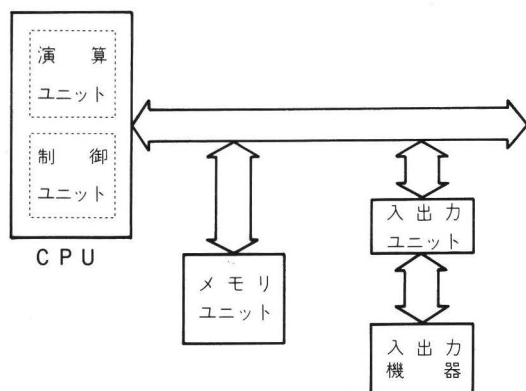
コンピュータの中身で何がもっとも重要かといえば，第1にCPU(Central Processing Unit,中央演算処理装置)です。CPUは計算したり，判断したり，全体をコントロールしたりするパソコンの中の指揮者のようなものです。

CPUは加減のような算術演算やAND, ORのような論理演算を行う「演算ユニット」と，パソコン全体の働きをコントロールする「制御ユニット」の2つの主要部分から構成されています(図I-3)。

CPUに高電圧の状態と，低電圧の状態を高速で繰り返す信号(クロック)が送られると，それに合わせて想像もつかないスピードで動作します。たとえばひとつの動作は1秒間になんと100万回も行われます。

CPUはLSI(大規模集積回路)の技術が進み，1チップに納まるようになりました。これをとくにマイクロプロセッサと呼びます。

X1シリーズでは「Z80」というポピュラーなマイクロプロセッサが使われています。

図I-3　コンピュータの基本構成

■メモリユニット—記憶装置

マイクロプロセッサ(CPU)は人間の頭脳に相当する重要な装置ですが，そのCPUにさせる仕事の手順やデータを蓄えておく場所がなければ，CPUは何の仕事もできません。その重要な役割をはたしているのがメモリユニット(記憶装置)です。

メモリユニットは，CPUにとって指令書であると同時に，計算ノートともいえます。メモリの内容を一見したところプログラムであるかデータであるかがわからないところに，

実は現在のコンピュータの奥深い問題がひそんでいるのです……。 CPUが実行する命令は機械語(マシン語)と呼ばれます。まえに説明したように，コンピュータは2進数の世界ですから，命令もやはり0と1の組み合わせになるのです。

CPUが実行する命令がマシン語ならば，BASICやPASCALなどのコンピュータ言語はいったい何なのだと思われるでしょう，たしかにパソコンはBASICで動きます。

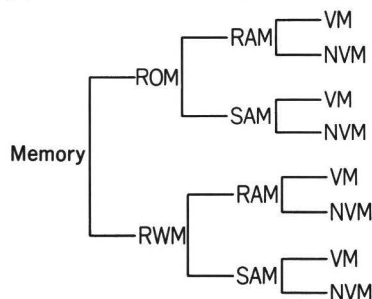
種明かしをしますと，CPUが直接BASICプログラムを実行しているのではなくて，BASIC言語をCPUがわかるようにマシン語に翻訳しながら実行しているのです。ソフトウェアがいかなる言語を使用していても，CPUがメモリから受け取る命令はいつもマシン語であるということを覚えておいてください。

メモリの分類

メモリは，3つの基準で分類することができます。

1．書き込むことができるかどうか
　読み出し専用メモリ(Read Only Memory:ROM)
　読み出し書き込みメモリ(Read/Write Memory:RWM)
2．メモリの読み出し(書き込み)が，ランダムかどうか
　ランダムアクセスメモリ(Random Access Memory:RAM)
　順序アクセスメモリ(Sequential Access Memory:SAM)
3．電源を切ると記憶内容が失われるかどうか
　不揮発性メモリ(Non Volatile Memory:NVM)
　揮発性メモリ(Volatile Memory:VM)

したがって，メモリの種類は計算上8種類(図I-4)となります。

図I-4　メモリの分類

ふつうよく使われるのは次の2種類です。

① **ROM-RAM-NVM**

② **RWM-RAM-VM**

よく「メモリはROMとRAMに分かれ……」といわれますが，①もRAM(つまりランダムアクセスできる)なので厳密には正しい表現とはいえません。

正確にはRWMとROMを区別すべきなのですがROM，RAMが広まってしまっているため本書でもそう呼びます。

ROM はさらに 2 種類に分かれます。

Ⓘ マスク ROM

Ⓘⓘ プログラマブル ROM(P-ROM)

メーカーが製作した記憶内容をⒾは書き換えることはできませんが，ロムイレーサとロムライタがあればⒾⓘは記憶内容を書き換えることができます。

メモリの容量

メモリの容量は「64 K ビット」などと書きます。この“K”はケイまたはキロと呼びますが，“1km”の k のように 1000 を表しているのではありません。

$1K=2^{10}=1024$

として使われているのです。

したがって 64K ビットのメモリということは，1024×64＝65536 個の 0 または 1 という情報（1 ビット）を記憶するだけの容量があることを示します。

ところで，任意の場所のデータを読み書きするためには，その場所を指定しなければなりません。このために各場所にアドレス（番地）をつけます。つまりメモリ中のデータ（1 ビット単位あるいは 8 ビット単位など）のすべてに番地をふるわけです。

CPU はメモリを読むときも書くときも，まず「何番地を読み書きしたい」とアドレスをメモリユニットに伝えます。

Z80 の場合，CPU チップの足のアドレスバスは 16 本，つまりアドレスの指定は 16 本のアドレスバスの 0 と 1 のパターンで行います。これも 2 進数ですが，便宜的に 16 進数で表します。

ここで 0000H 番地～0FFFH 番地まで，つまり 1000H 番地分のメモリの容量を計算してみましょう。

16 進数の 1000 を 10 進数に直すには，

$16\times16\times16\times1+16\times16\times0+16\times0+0=4096$

とします。これで 4096 番地分の容量であることがわかりました。Z80CPU は 1 番地に 1 バイトのデータを扱うので，結局，4096 バイトであるということになります。また，

$4096=1024\times4$

ですから 4KB（バイト）ともいえるわけです。

ここで 1K＝1024 とした便利さがわかっていただけたでしょう。2 進数からすると 1000 より 1024 のほうがきりのいい数なのです。

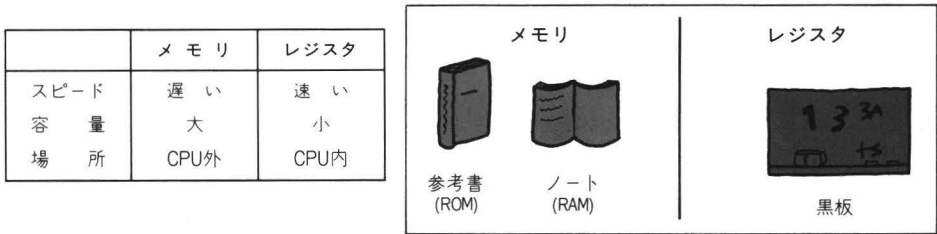
■レジスタ

マイクロプロセッサ（CPU）の中にレジスタと呼ばれる部分があります。ここは計算結果などのデータを一時蓄えておくところです。といってもひとつのレジスタの中には 1 バイト（0，1 の 8 個の組み合わせ）か 2 バイトぐらいしかデータを置くことができません。

レジスタの数は 10～20 ぐらいあり，それぞれ名前がついています。レジスタはプロセッサ内にあるので，アドレスバスやデータバスを使ってデータを転送するメモリユニットよ

りもずっと高速です。このため，レジスタは計算の途中結果などを一時的に保管するのに使っています(図Ｉ-5)。

図Ｉ-5 メモリとレジスタ

	メモリ	レジスタ
スピード	遅い	速い
容量	大	小
場所	CPU外	CPU内

■入出力装置

コンピュータはCPUとメモリユニットだけでは働くことができません。コンピュータに指示やデータを与える入力装置，結果を取り出すための出力装置が必要です。

入力装置にはいろいろなものがありますが，もっとも一般的なものはキーボードです。機械に通じる言葉(コンピュータ言語やコマンド)を打ち込むためのさまざまな文字キー，数字キー，機能キーがついています。

このほか，最近では音や声による入力の研究が進んで，音響カプラや音声入力装置が開発されています。さらに画面指示用のライトペン，画面の図形位置を自由に移動させられるジョイスティックやマウスなど，入力方法の可能性はぐんと広がっています。

コンピュータのした仕事の結果を人間にわかる形で取り出す装置，それが出力装置です。パソコン利用の目的が多様であるだけ，出力装置も多様なものがあります。その出力装置の代表的なものがCRT(ブラウン管)とプリンタです。

ここで入力にも出力にも使われる外部記憶装置にふれなければなりません。コンピュータを動かすためのプログラムのほか，処理するデータや処理ずみデータを格納するのが外部記憶装置です。

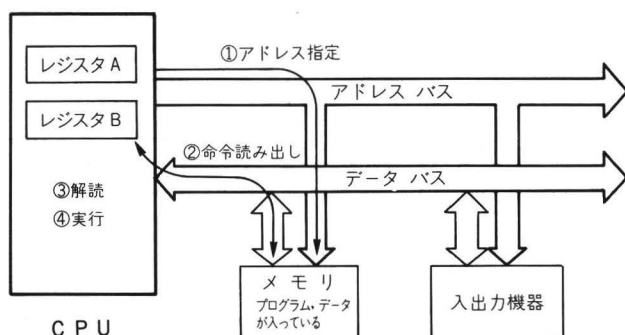
外部記憶装置としてもっとも手軽なのはカセットレコーダです。ただしこれは必要な部分を取り出すのに時間と手間がかかります。この欠点をカバーするのがフロッピディスクです。X1シリーズでは両者のタイプが用意されています。そのほかにはMZ-1500に内蔵されている簡易版といえるクイックディスク(QD)や，より大きなシステムにする際に欠かせないハードディスクがあります。

コンピュータの動作

■動作手順

コンピュータとはいったいどんな手順で動作しているのでしょうか。コンピュータの秀れた性能からすると，よほど複雑なことをしているように思われますが，意外と単純なのです。

その手順を知れば「何だ，コンピュータってこんなものか」と思う人もいるでしょう。ではコンピュータはどのように動作しているのか，その一瞬を取り出しスローモーション(約1000万倍減速)でみてみましょう(図I-6)。

図I－6　コンピュータの動作手順

① コンピュータに仕事をさせるためのプログラム(指令書)がメモリに順に記憶されています。そこでまずメモリからその指令をひとつずつ取り出さなければなりません。マイクロプロセッサ(CPU)は「メモリの＊＊＊＊番地の命令をもってきなさい」とメモリに指示します。

② メモリは，指定された番地にある命令(マシン語)を取り出しCPUに差し出します。

③ CPUは命令の意味を解読します。

④ いよいよ命令の実行です。実行する動作のプロセスは，おもに次のようなものがあります。

ⓐ メモリ↔CPU，入出力装置↔CPU，CPU(レジスタA)↔CPU(レジスタB)の間などでデータを移動する。

ⓑ CPU内で加減などの計算をする。たとえばレジスタBの内容をレジスタAに加算し，結果をAに置く。

ⓒ 通常，若い番地から順に命令をひとつずつ実行するが，離れた番地から実行するようにコントロールの流れを変える。

実行は④から再び①へ移り，①→②→③→④を繰り返してゆくのがコンピュータの手順です。①と②を合わせて命令フェッチ，③をデコード，④を実行と呼びます。

仕事のやり方は，このように単純なのですが，ソフトウェアや入出力装置によって，コンピュータはとても素晴らしい仕事を実行するのです。

マイクロプロセッサや各IC類は連絡を取りながら仕事を進めます。うまく歩調を合わせるために各部分がひとつのリズムに乗って動作しています。これがクロックと呼ばれているものです。

■3つのCPU

あるひとつの仕事が与えられたとき多くの人が力を合わせた方がスピードが上がります。理想的にはn人いれば1人でやるより1/nの時間ですむはずです。コンピュータでも同様でn個のマイクロプロセッサを装備すればスピードがn倍になるはずです。とはいうもののおもにソフトウェアの問題(そもそもふつうのプログラムに並列性がない)から，なかなかうまくゆきません。

X1シリーズでは限定された形ながら，機能分散型のマルチプロセッサ構成をとっていて，メインCPUの他にふたつのサブCPUが活躍しています。ただしサブCPUは自由にプログラミングできず，メインCPUに従属しています。

先進のハードウェアX1の構成

■CPU Z80A

ここでは，システムダイアグラムと基板写真を見ながら，X1およびX1 turboの各ユニットの働きを具体的なイメージとしてつかむことにしましょう(図I-7, 8)。

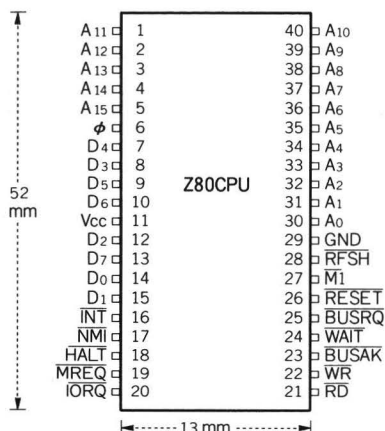
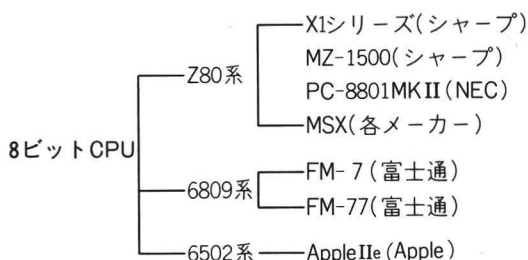
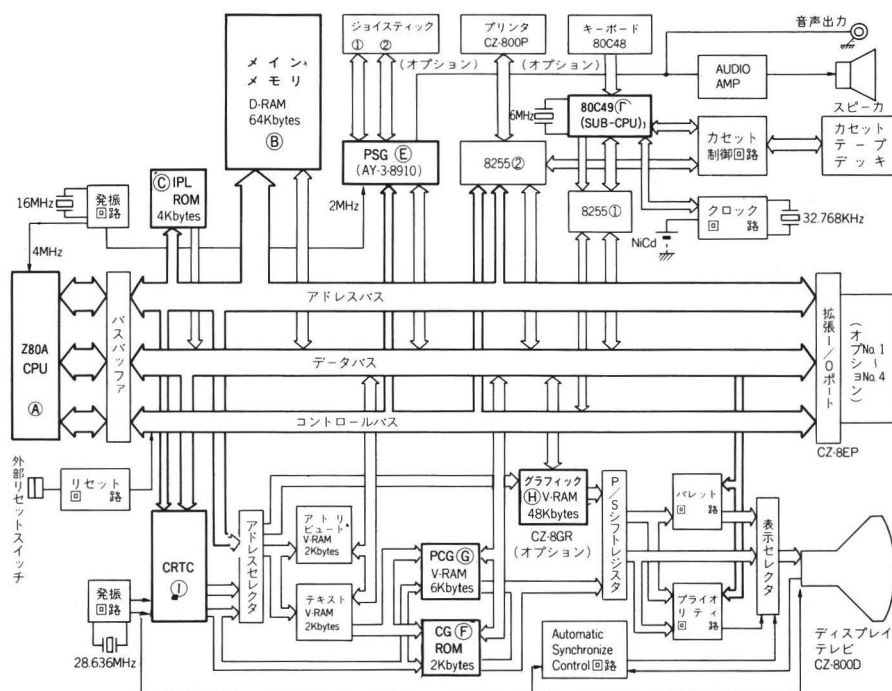
X1の中心的な役割をになっているCPUには，ポピュラーな8ビットマイクロプロセッサ「Z80A・写真Ⓐ」が使われています(図I-9)。

CPUを動かすには，前に説明したように，高電圧と低電圧をすさまじいスピードで繰り返す信号，つまりクロックをつながなければなりません。Z80Aの“A”というのはクロック4MHzに対応するバージョンであることを表しています。1秒間に400万回というわけです。

これに対してAがついていないZ80バージョンは，2MHzのクロックまでしか保証されません。

Z80(あるいはそれに相当するCPU)を装備しているパソコンは，X1シリーズ以外にも数えきれないほどあります。コンピュータの世界は，とにかく早く普及させたものが勝ちなのです。これはおもに①ソフトウェア資産の継承(いわゆるコンパチ)が重要であること，②普及すればするほど加速度的に安くなることに理由があります。

一方，Z80以外の有力な8ビットCPUとしては，富士通のFMシリーズに採用されている6809や米アップル社のApple IIが採用している6502などですが，これらもスッキリと命令体系が整っていてなかなか秀れています。むしろ熱狂的なファンはこちらの系統の方が多いともいえます(図I-10)。

図I-9　Z80CPU端子配置図

図I-10　8ビットCPUと代表機種

図Ⅰ-7　X1システムダイアグラム

■メインメモリ

写真Ⓑがメインメモリです。全体で64Kバイトの容量を持っています。Z80Aからアドレス線が16本出ていますから，表すことのできる番地は次のとおりです。

$0000000000000000_{(2)} \sim 1111111111111111_{(2)}$

つまり16進数では，

0000H～FFFFH

これは64Kバイト($=2^6 \times 2^{10}$)に相当します(ひとつの番地には1ビットではなく1バイト格納することができます)。すなわちCPUに直接つなげる最大量のメモリが接続されているということです。

メモリチップが8個並んでいるのが写真からわかります。これはデータ線1本ずつに対応します(図Ⅰ-11)。

図Ⅰ-11　メインメモリ容量

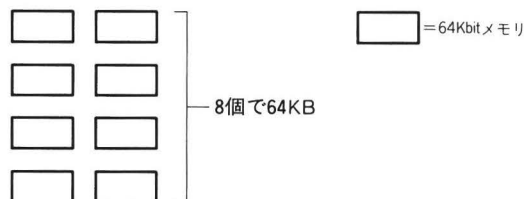



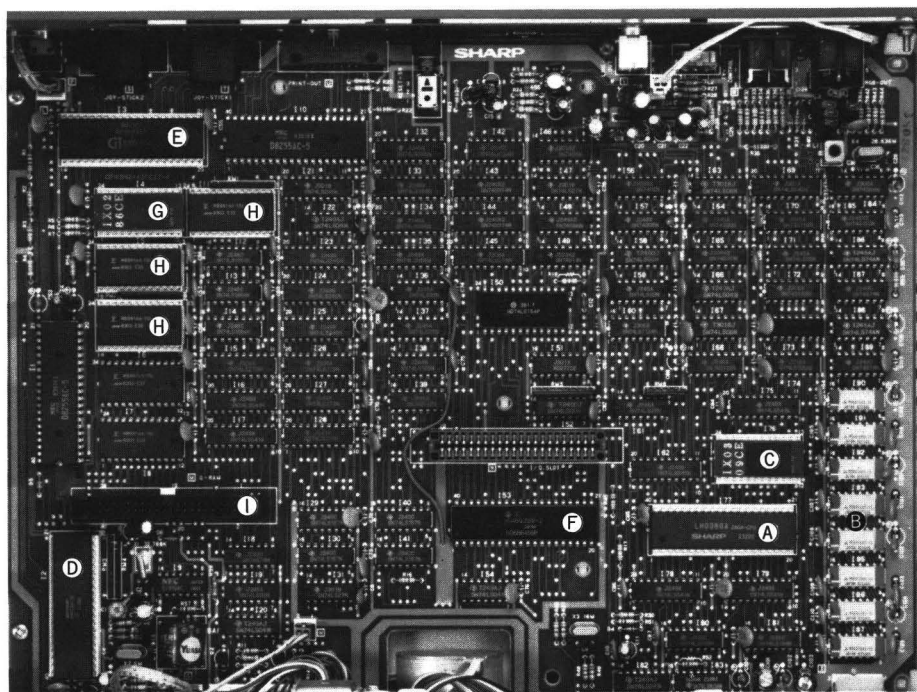
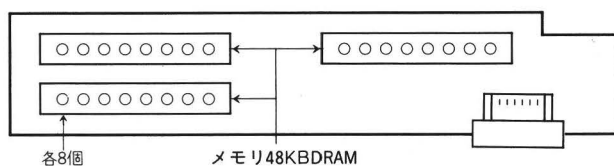
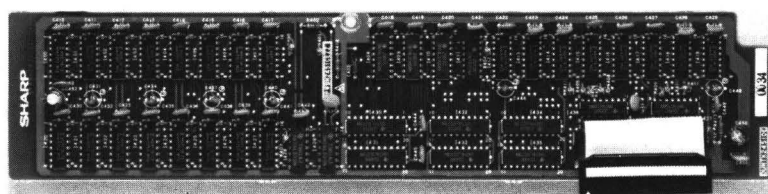
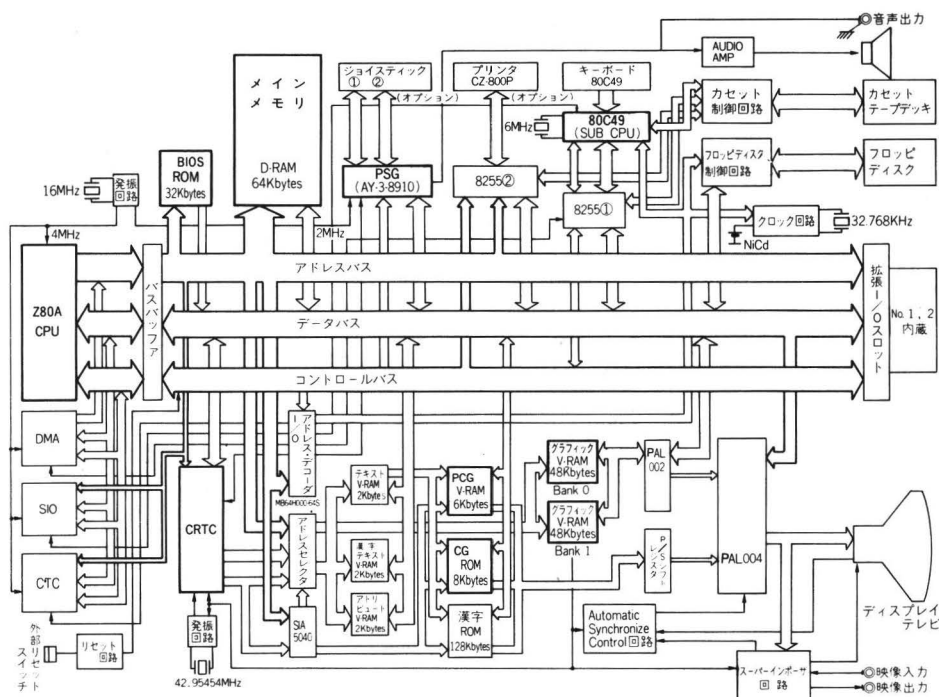
Photo I-1　X1メインボード

図I-12　グラフィックV-RAMボード

注）V-RAMボードのセットは，メインボードのコネクタ（写真Ⓘ）に挿入する

Photo I-2　GRAPHICボード

1チップの容量が64Kビットですから，

　64Kビット×8＝64Kバイト

となるのです。

なお，メインメモリは，電源を切ると内容が消えてしまうので，電源を入れた直後にカセットやディスクからプログラムをメインメモリに読み込んでくる（転送してくる）プログ

図Ⅰ-8　X1 turboシステムダイアグラム

ラムが必要です。X1では，カレンダタイマー機能なども持たせたROM(IPL ROM・写真©)が装備されています。X1 turboは，turbo以外のX1ではテープやディスケットなどによって読み込まれるBASICの一部(IOCS部)もROMの中に含めてBIOS-ROMとしています。容量も4Kバイトから32Kバイトとぐんと大きくなっています。

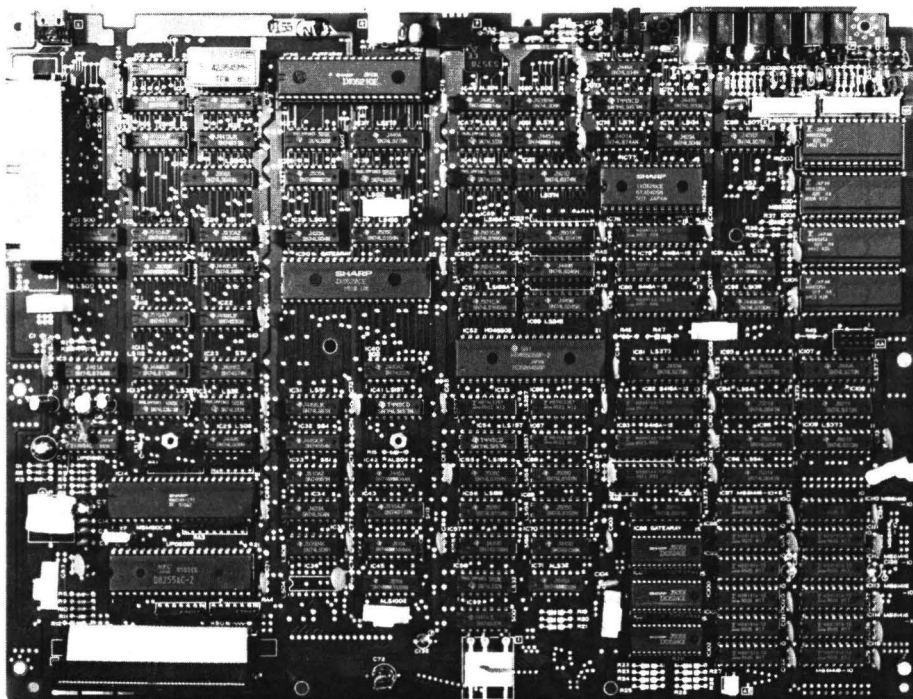
電源を入れるとカセットテープ(あるいはディスケットまたはROM)からBASICインタプリタをメインメモリに転送し，それを実行します。ただし，メインメモリにあるインタプリタ(あるいはその他のソフトウェア)実行中はIPL ROMはじゃまになりますから，電気的に切り離しておきます。

■サブCPU

X1には，メインCPU(Z80A)のほかに，入出力用のCPUが2個搭載されています。このCPUは，ユーザーがプログラミングすることはできません。メインプロセッサが必要に応じてサブCPUにコマンドを送るという形式をとります。

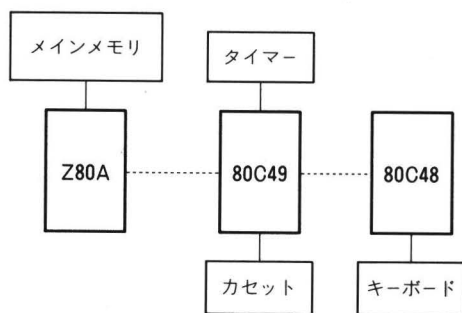
まず80C48(X1 turboでは80C49)は，CPUの本体中ではなくキーボード部にあり，押されたキーのデータをシリアル(直列)に本体部に送る働きをします(図Ⅰ-13)。

もうひとつの80C49というCPU(写真Ⓓ)は，80C48(80C49)から送られたキーデータを受け取ります。さらに，カセット装置やテレビのコントロール，タイマー回路のコントロールも行います。

Photo I -3　X1 turboメインボード

ところで，このサブCPUの80C49は，メインCPUのZ80Aから完全に独立して別のことを行うわけではありません。メインCPUの指示に従って仕事をしたり，メインCPUにデータを送ったりするのです。そこで両者間のデータ送受信のために，8255というパラレル(並列)インタフェース素子が使われています。これはデータの往き来を調整するものと考えればよいでしょう。低電圧と高電圧が衝突することは，基本的なハードウェアのバグに属します。

カセットやプリンタとのインタフェースに，もうひとつ8255が使われています。

なおX1とX1Cに装備されているカセット装置は，信頼性が高く秀れた機能を持つフル

図I -13　3つのCPUで機能分散

ロジックメカニズムタイプですが，これもサブCPU 80C49によるところが大きいといえるでしょう。

■PSG－サウンド発生用IC

写真Ⓔにあたる PSG(プログラマブルサウンドジェネレータ)は，AY-3-8910というチップです。これは3つのサウンドの音程と音量を設定して3重和音を出すことができ，しかもそれにノイズを加えることができるものです。そして全体の音量を周期的に変化させることもできます。

このチップはホビー向けパソコンによく登場するので，8ビットパソコンの統一規格MSXでも採用されています。

PSGをコントロールする命令はBASICにもあり，音程の計算などを行えば簡単に使うことができます。

なお，このチップにはPSG機能とは別に，自由に使えるポート(外部機器との接続の出入り口)があり，X1ではジョイスティックがふたつまで接続できるようになっています。

■画面周辺

画面周辺を見ていきましょう。

X1本体にある画面のデータを，タイミングをはかりながらTVディスプレイにシリアルに送るために，46505というCRTコントローラ(CRTC・写真Ⓕ)を使っています。機能的にはシンプルですが，使いやすくいろいろ応用できるので広く普及しています。

このCRTCを使って，CRTの水平垂直同期信号の発生や，キャラクタとグラフィックのアドレスの発生，表示タイミングのコントロールを行っています。

画面に表示される『A』『！』などのキャラクタのドットパターン(フォント)の情報は，キャラクタジェネレータ(CG)というROM(写真Ⓖ)に記憶されています。1キャラクタにつき8バイト必要で，キャラクタは256個あるのでCG ROMは，

　　8×256＝2Kバイト

の容量を持ちます。

X1 turboは，高解像度モードと低解像度モードの2種類のフォントを4Kバイトずつ計8Kバイトの容量を持っています。さらに漢字ROMが128Kバイトも標準装備されています。

文字パターンをユーザーが自由に定義できるようにするため，PCG(プログラマブルCG)が装備されています(写真Ⓗ)。この場合は書き込みもできるメモリでなくてはならないので，RAMで赤(レッド)，青(ブルー)，緑(グリーン)の3色に対応してそれぞれ2Kバイト，計6Kバイトの容量があります。

画面のどこにどういう文字が入っているかという情報を持つのがテキストV-RAM 2Kバイトです。さらに，それぞれの文字がどういう属性(CGROMかCGRAMか，あるいは色や大きさなどの性質)を持つのかという情報は，アトリビュートV-RAM 2Kバイトに保持されます。

X1 turboは漢字表示用のテキストV-RAM 2Kバイトを別に持っています。

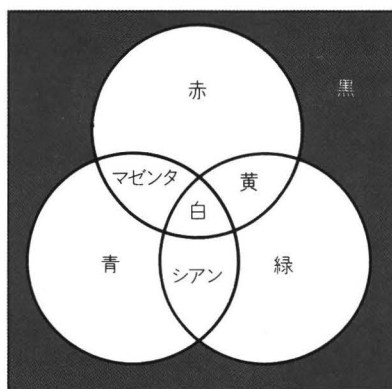
画面にドット単位に点を打ち，図形を描くためのグラフィックV-RAMボードを挿入するコネクタ(写真①)があります。レッド，ブルー，グリーンの16Kバイトずつ計48Kバイト(turboは2倍の96Kバイト)です。この3色の重ね合わせによって，図1-14のような8色を出すことができます。

多様性のあるグラフィックスを実現させるためには，画面データをそのままディスプレイに送るのではなく，間にプライオリティ回路とパレット回路を付けます。これはX1シリーズの最大の特徴といってもいいでしょう。

図Ⅰ-14　8カラー発色図

プライオリティ機能は，グラフィックスの各色8色とキャラクタの間に順番をつけ，その優先順位に応じて画面に表示するものです。これによって遠近効果を出すことができるようになりました。

パレット機能は，実際に出力する色のデータを，プログラムで設定することができるもので一瞬のうちに色を変化させます。

また，ASC(Automatic Synchronize Control)回路で，スーパーインポーズが実現されます。これはTV画像やビデオ画像などとコンピュータの画像を合わせて表示するものです。

■X1シリーズの流れ

1982年秋，次々とMZ系列の新機種を送り出していたシャープは突如として新しいコンセプトを持ったX1(CZ-800C)を発表しました。

パソコンの性能をスピードとメモリ容量のふたつのものさしで評価しがちなユーザーに，この「使い込みたくなるマシン」は大きなショックを与えたものでした。勢いにのって基本部分は同一のX1C(CZ-801C)，X1D(CZ-802C)を発売し，その後X1Cは，X1CS/K(CZ-803C/804C)の2タイプそしてX1Fにひきつがれました。

ここまでの系列機種は，市場原理からしても当然のラインナップといえるでしょう。

ところが元祖X1の登場から約2年後，再び革新的ともいえるX1 turboが登場し，旋風を呼び起しています。

グラフィックス，日本語処理，ビデオとの融合など，今までの8ビット機種では考えられない性能をもちながら，しかも従来のX1シリーズとの「完全上位コンパチ」を実現したのでした。

前出のX1およびX1 turboのシステムダイアグラム図を比較しただけでも，次のような大幅な性能向上がわかります。

1．IPL ROM(4Kバイト)がBIOS ROM(32Kバイト)に拡大した。

2．キーボード部のサブCPUが80C48から80C49に強化された。

3. Z80 DMA, SIO などZ80ファミリのLSIが付加された。

4. 漢字用のCGとV-RAMが付加された。

5. グラフィックV-RAMの容量が2倍になった。

6. スーパーインポーザを内蔵した。

7. CG-ROMの容量が2Kバイトから8Kバイトに拡大した。

これらの性能向上に加えて, ただでさえ強力なHu-BASICがさらにパワーアップ(とくに日本語処理面)し, 最強のものとなりました。

なお, X1 turboにもいくつか種類がありますが, 本書では圧倒的に普及していると思われるディスク内蔵タイプ(CZ-852Cなど)を想定しています。

column2

メモリの階層構造

よいメモリとはどういうものでしょうか。それはまずアクセス(読み書き)が速いことです。最近はマイクロプロセッサ自体はどんどん性能が上昇しており, スピードネックは主メモリへのアクセスという場合が多いからです。

ところがスピードが速いメモリは, 値段が高くなりますし集積度を大きくすることも難しいものです。したがってコストも含めてパフォーマンスを上げるための方法がメモリの階層化です。

通常のパソコンもメインメモリと外部メモリ(たとえばディスクなど)でレベルを分けていますが, もう少し高級なコンピュータではメインメモリよりも速く, 容量が小さいというモジュール(キャッシュメモリ)を使っています。

一方プログラムにとって, この階層化は面倒くさいものといえます。そのためにハードウェアとオペレーティングシステムでこの部分をおおってしまいプログラマがまったく物理構造を意識しないですむようにします。これがいわゆる仮想記憶です。

II X1ブラックボックスを探検する —1

●

マイクロプロセッサとマシン語

Z80の構成と命令体系

アセンブリ言語について

メモリ空間とI/O空間

メモリ空間とI/O空間の考え方

メモリ空間の構成

I/O空間の制御

マイクロプロセッサとマシン語

Z80の構成と命令体系

■Z80の構成

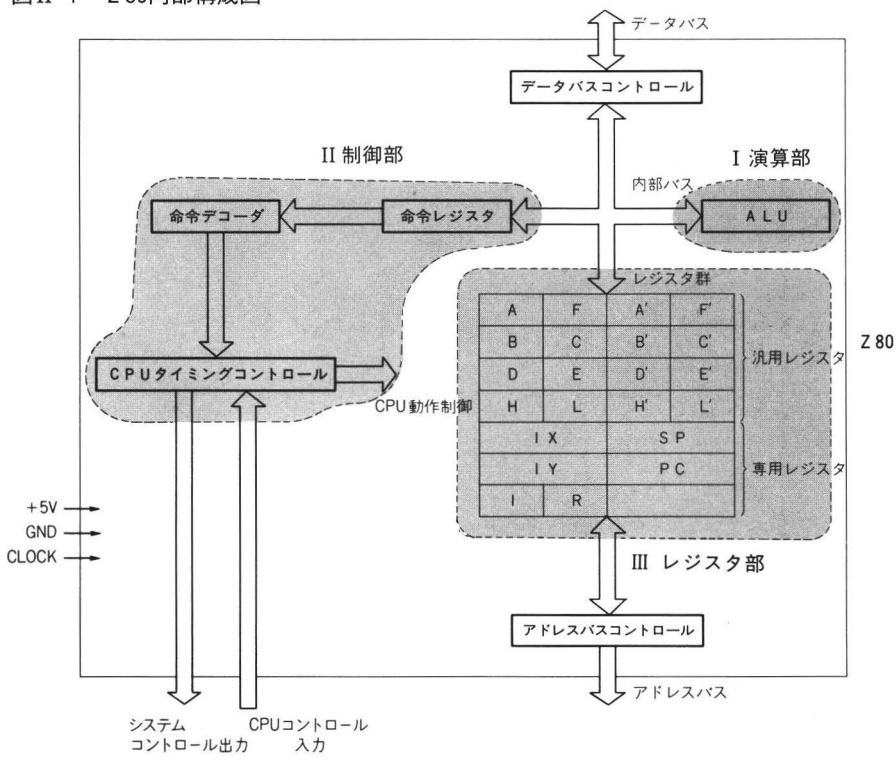
CPU Z80の内部構成を表したのが図II-1です。機能的には次の3つの主要部分から成り立っています。

I 演算部

II 制御部

III レジスタ部

演算部は加減算などの算術演算とAND,ORなどの論理演算を実行し，制御部は命令に応じてCPUの動作のコントロールを行います。レジスタ部にはいろいろなレジスタが含まれており，汎用レジスタと専用レジスタの2種類に分けることができます。各レジスタは使用上の特性を持っており，この特性は命令を使っているうちにしだいにわかってくる

図II-1　Z80内部構成図

ものです。よく使うレジスタはA,B,C,D,E,F,H,Lでそれぞれ8ビットです。BC, DE, HLはペアレジスタを構成し，16ビットのレジスタとしても使うことができます。

CPUの動作を決定するのが「命令」です。命令は命令レジスタに入り解読され，それに応じて演算論理ユニット(ALU)を動作させたり，データを移動させたりします。たとえば，

01000111(=47H)

という命令により，

B←A

つまりレジスタAの内容がレジスタBの中に入ります。今の例は1バイトの命令でしたが，2バイトの命令もあります。

00111110(=3EH)
10001010(=8AH)

という命令によって，

A←8AH

つまりレジスタAの内容が8AHとなります。同様に3バイト，4バイトの命令もあります。

命令をいくつか組み合わせて少しプログラムらしきものを作ってみましょう。

B000番地とB001番地の内容を足してB002番地へ入れる

この実行がひとつの命令でできれば楽なのですが，残念ながらZ80では加減算はレジスタAを介さなければならないのです。そこでたとえば次のような順序で命令を実行すればよいでしょう。

```
A←(B000H)ⓐ
B←A      ⓑ
A←(B001H)ⓒ
A←A+B    ⓓ
(B002H)←Aⓔ
```

このプログラムは次のようになります。

```
00111010   (=3AH) ┐
00000000   (=00H) ├ ⓐ
10110000   (=B0H) ┘
01000111   (=47H)   ⓑ
00111010   (=3AH) ┐
00000001   (=01H) ├ ⓒ
10110000   (=B0H) ┘
10000000   (=80H)   ⓓ
00110010   (=32H) ┐
00000010   (=02H) ├ ⓔ
10110000   (=B0H) ┘
```

このような0と1の数字の列が命令でありそれがマシン語そのものを構成しています。

マシン語は人間にとっては理解のしにくいものといえます。このために用意されているのがアセンブリ言語です。これはマシン語命令ひとつにひとつずつ対応してアセンブリ命令が作られており，プログラミングがわかりやすいようにアルファベットを用いて意味を持たせたものです。

あなたがアメリカで生まれたのならばもう少し意味がよくわかったでしょう。

アセンブリ言語自体をCPUは理解できませんので，アセンブリ言語を機械語に翻訳するステップが必要です。対照表を見ながら人間がひとつずつ変換していくことも可能ですが，大きなプログラムになってくるとコンピュータにまかせなくてはならなくなってきます。そのような変換をするためのプログラムをアセンブラといいます(図II-2)。

プログラム例で示したものをアセンブリ言語で表してみると次のようになります。

リストII-1

```
LD      A,(0B000H)  ⓐ
LD      B,A         ⓑ
LD      A,(0B001H)  ⓒ
ADD     A,B         ⓓ
LD      (0B002),A   ⓔ
```

注) B000Hの前の0はつけなくてもよい場合があります

アセンブリ言語はマシン語に1対1対応しているものですから，CPUが違うと当然違ったものとなってしまいます。この点が同じ「言語」でもBASICやCなどの高級言語と大きく異なっているといえます。具体的にZ80の内容をみていくことにしましょう。

図II-2　アセンブラの働き

汎用レジスタ

Aレジスタ

アキュムレータとも呼ばれ，もっとも重要な働きをします。8ビットの加減算，論理演算，比較はすべてAレジスタの内容に対して行われます。

Fレジスタ

Flagレジスタとも呼ばれ，このレジスタは各ビットごとにちがった意味を持ちます。いずれも8ビットあるいは16ビットの演算結果の状態が格納されます。演算の結果に応じてプログラムの流れを制御するときに参照するためのものです。図II-3にFレジスタビット構成を示します。

図II-3　Fレジスタのビット構成

Fレジスタ

第7ビット　　　　　　　　第0ビット

S	Z	−	H	−	P/V	N	C

(−：未定義)

1. Cフラグ　Aレジスタの最上位ビットからの桁上がりでセット(1になる)されます。
2. Nフラグ　先に実行された命令が加算か減算かの判定時に参照します。減算の場合はセットされます。
3. P/Vフラグ　このフラグはふたつの機能を持ち，論理演算が行われた場合にはパリティを示しパリティが偶数ならばセットされ，奇数ならばリセットされます。符号付の補数値の演算が行われた場合にはオーバーフローの有無を示し，オーバーフローがあ

った場合はセットされ，なかった場合はリセットされます。

4. Hフラグ(ハーフキャリーフラグ)　BCD演算の結果の下位4ビットからのキャリー，ボローの有無を示し，結果にキャリー，ボローがあればセットされます。

5. Zフラグ(ゼロフラグ)　演算の結果，レジスタに0が格納されたときにセットされます。

6. Sフラグ(サインフラグ)　符号付数値演算の結果が負ならばセットされます。

これらのフラグのうちC，Z，S，P/ Vの4つのフラグは 参照 のみではなく，プログラムにより操作することもできます。

B, C, D, E, H, Lレジスタ

この6つのレジスタは汎用として使用でき，また2本のレジスタをペアにしてBC, DE, HLの16ビットレジスタとしても使えます。これらの汎用レジスタは，まったく同等に使われているのではなく，各命令によって使用されるレジスタが異なる場合があります。

たとえばBレジスタは，ループの実行制御命令と組み合わせて使われたり，Cレジスタは入出力命令に使用されることがあります。このような各レジスタの性格は，小さなプログラムを作っていくうちにしだいにわかってくるものです。

今まで述べたA，F，B，C，D，E，H，Lレジスタには予備のためにもうひとつずつ(A',F',B'……)レジスタが用意されており，命令によってお互いに内容を一度に交換して使用することができます。

専用レジスタ

PCレジスタ

プログラムカウンタレジスタは，CPUが次に実行すべきマシン語のメモリアドレス(16ビット)を保持しており，メモリからその次の命令を読み出すときに参照されます。

SPレジスタ（スタックポインタ）

外部RAM上のスタック領域のトップアドレス16ビットを保持するものです。スタック領域とは レジスタの内容，サブルーチンコールからリターンするためのメモリアドレスなどを一時的に記憶するためのメモリ領域です。スタックへのデータの一時記憶はLIFO(Last In First Out)方式で行われます。つまり一番最後にスタックに記憶したデータが一番最初に取り出され，SPレジスタはメモリの小さい方へ進み値が減っていきます。

IX, IYレジスタ

メモリのある場所を指定する方法として，直接番地で表す方法やあるレジスタの中身を番地として表す方法などがあります。その他の方法として，相対アドレッシングという方法があります。これはある基準とする番地を一時的に固定しておいて，その番地からの差分によって指定するものです。IX, IYレジスタは，メモリアドレスの相対的アドレッシングのために16ビットの基準アドレスを保持するレジスタです。2個独立に使用できデータ

テーブルを参照するような場合などによく使用します。

Iレジスタ　(割り込みベクトルレジスタ)

Z80 CPUの割り込みモード2で使用します。X1でもこの割り込みモードをキー入力ルーチンなどに使っています。この方式によれば割り込み処理ルーチンをメモリのどこへでも配置できZ80 CPUのひとつの特徴となっています。

Rレジスタ(メモリリフレッシュレジスタ)

ダイナミックメモリのリフレッシュに使われ，プログラムには直接の関わりはあまりありません。

■Z80の命令体系

CPUのアーキテクチャ，設計思想の良悪を特徴づけるのは何といってもその命令体系です。Z80 CPUには158種の命令があり，それぞれ機能に応じて1～4バイトで構成されています。プログラムを作るときにはこれらの命令の働きを理解し，アルゴリズムの流れをそのCPUの命令体系で表現できるように考える必要があります。

ところでCPUが直接解読できるのはもちろん1または0を8個組み合わせた1バイトの記号(列)ですが，プログラミングが楽になるために各命令にその命令に関する人間がわかりやすい言葉がついており，これをニモニックと呼びます。たとえば，

01000111　　(＝47H)

という命令はAレジスタの内容をBレジスタにコピーする働きをするものですが，これをニモニックでは次にように書きます。

```
LD      B,A
```

LDはこの命令の動作(ロード，LoaD)を表します。このような命令の動作を示す部分をオペコード(Op.code)と呼びます。一方B，Aは命令の動作の対象となるものを示す部分でオペランド(Operand)と呼びます。命令によってはオペランド部のないものもあります。

このようにマシン語コードをニモニックで表した言語体系をアセンブリ言語と呼びます。マシン語とアセンブリ言語は完全に1対1に対応しています。したがってアセンブリ言語はCPUごとに異なります。しかしひとつのCPUについて学べばほかのCPUのアセンブリ言語も比較的スムーズに理解できるものです。

Z80 CPUの命令群は大別して次のようなグループに分けられます。

・データの移動，交換

・ブロック転送，ブロック・サーチ

・算術，論理演算

・ローテイト，シフト

・ビット操作

・ジャンプ，コール，リターン

・入力，出力

・CPU 制御

ここではZ80のこれらの各命令について説明し，命令のニモニックと同時に対応するマシン語コードを示します。以下の命令表の数字はマシン語コードを示し，記号については次のように定義します。なお，これらの命令はZ80命令の詳細ですので，とくに詳しく知りたい方は注意してお読みください。

d：－128～＋127(符号付バイトの数値,IX,IYレジスタからのディスプレイメント・差分)

n：00H～FFH(1バイトの数値)

nn：0000H～FFFFH(2バイトの数値)

b：ビット(b＝7は第7bitを示す)

e：－128～＋127(相対ジャンプ，コールのディスプレイスメント，次の命令の先頭番地を0とする)

ロード命令

ロード命令はデータとCPUのレジスタ間で，あるいはCPUのレジスタと外部メモリ間などで，データを転送する機能を持っています。この命令ではデータを取り出すソースとそれを格納するデスティネーションを指定しなければなりません。転送といってもソースの内容は変化しないのでコピーといった方がよいかも知れません。

ロード命令には8ビットロードと16ビットロードのものがあり，図II-4に8ビットロード命令グループを示します。データのソースは上欄に，デスティネーションは左側に示

図II-4　8ビットロード命令グループ

ソース

デスティネーション			インプライド		レジスタ							レジスタ間接			インデックスド		拡張アドレッシング	イミディエット
		X / ニモニック	I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX +d)	(IY +d)	(nn)	n
レジスタ	A	LD A, X	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B	LD B, X			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C	LD C , X			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D	LD D, X			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E	LD E, X			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H	LD H, X			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L	LD L, X			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
レジスタ間接	(HL)	LD(HL), X			77	70	71	72	73	74	75							6 n
	(BC)	LD(BC), X			02													
	(DE)	LD(DE), X			12													
インデックスド	(IX+d)	LD(X+d), X			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)	LD(IY+d), X			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
拡張アドレッシング	(nn)	LD(nn), X			32 n n													
インプライド	I	LD I, X			ED 47													
	R	LD R, X			ED 4F													

してあります。この命令グループのニモニックオペコードは，

LD

であり，これのあとにオペランドをデスティネーション，ソースの順で書きます。たとえばBレジスタの内容をAレジスタにロードする命令は，

LD　　　A,B

と書き，図II-4の表よりこれに対応するマシン語は，

78H

となります。

一般に，命令の操作の対象となるデータの所在場所の指定方法をアドレッシングと呼びます。Z80CPUにも種々の方式が用意されており，その選択によってよりよいプログラムができます。図II-4の上方と左側に示しているのは8ビットロード命令グループにおけるアドレッシングモードを表しています。

LD　　　B,C

におけるCレジスタの指定は「レジスタ直接アドレッシング」といい，レジスタの内容を操作対象とします。

LD　　　A,(HL)

の『(HL)』の指定は「レジスタ間接アドレッシング」といい，16ビットレジスタの内容をアド

図II-5　16ビットロード命令グループ

		ソース							
		レジスタ						拡張イミディエット	拡張アドレッシング
	ニモニック＼X	BC	DE	HL	SP	IX	IY	nn	(nn)
デスティネーション レジスタ BC	LD BC, X							01 n n	ED 4B n n
デスティネーション レジスタ DE	LD DE, X							11 n n	ED 5B n n
デスティネーション レジスタ HL	LD HL, X							21 n n	2A n n
デスティネーション レジスタ SP	LD SP, X			F9		DD F9	FD F9	31 n n	ED 7B n n
デスティネーション レジスタ IX	LD IX, X							DD 21 n n	DD 2A n n
デスティネーション レジスタ IY	LD IY, X							FD 21 n n	FD 2A n n
デスティネーション 拡張アドレッシング (nn)	LD (nn), X	ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n		

ニモニック＼X	AF	BC	DE	HL	IX	IY
PUSH X	F5	C5	D5	E5	DD E5	FD E5
POP X	F1	C1	D1	E1	DD E1	FD E1

レス番地とするメモリの内容が操作対象となります。

```
LD      A,(IX+d)
```

の(IX+d)の指定は「インデックスアドレッシング」といい，16ビットインデックスレジスタとディスプレイスメント(d)との組み合わせで示したメモリアドレスの内容が操作の対象となります。

```
LD      A,n
```

のnの指定は，「イミディエットアドレッシング」といい，オペコードに続く1バイトが操作の対象となります。

```
LD      A,(nn)
```

の『(nn)』の指定は「拡張アドレッシング」といい，メモリアドレスの内容が操作の対象となります。

```
LD      A,I
LD      A,R
```

のIとRの指定は「インプライドアドレッシング」といい，このモードではオペコード自体がレジスタ指定を含んでいて，ひとつまたはふたつのレジスタが自動的に指定されます。

図II-5に16ビットロード命令グループを示します。16ビットロード命令グループでは，

```
LD      BC,nn
```

のようなソース側(nn)に拡張イミディエットアドレッシングの指定ができます。このモードではオペコードに続く2バイトが操作の対象となります。

PUSH，POP命令

PUSH, POP命令も一種の16ビットの転送命令で，図II-5の16ビットロード命令と一緒に示してあります。これらの命令のニモニックはそれぞれ，

```
PUSH  ソース
POP   デスティネーション
```

で，16ビットペアレジスタとスタックポインタ(SP)レジスタが指定した外部メモリ(スタック)間との16ビットのデータ転送命令ですが，ただこのときはスタックポインタの増加(POP)あるいは減少(PUSH)が自動的に行われます。前に述べたようにスタックへのデータのPUSHとPOPはLIFO方式で，最後にスタックされたものが最初に取り出されます。たとえばSPの内容が，1234Hのときに次の命令を行うと，

図II-6　PUSH BC

```
PUSH      BC
```

により，1234H番地にBレジスタの値，1233HアドレスにCレジスタの値が格納されたあとSPの値は自動的にデクリメントされ2だけ小さくなります(図II-6)。

POP命令はこれと反対の動作となります。

したがってPUSH命令につづいてPOP命令を行うとSPレジスタは変化しません。

POP時のレジスタペアの指定はPUSH時と必ずしも同じでなくてもかまいません。逆にこれを利用してZ80命令にはないような16ビットレジスタ間のデータ転送を行う方法がよく使われます。たとえば,

```
PUSH     BC
POP      DE
```

としますと，DEレジスタにBCレジスタの値がコピーされます。

交換命令

交換命令はふたつの16ビットレジスタのデータを入れ換えます。図II-7に交換命令グループを示します。この命令グループのニモニックのオペコードは,

EX と EXX

で，図II-7の6個の命令があります。

ブロック転送命令

ブロック転送命令はある連続したメモリ領域からデータをほかの領域にまとまって転送する機能をもちます。ひとつの命令で実現されるのですからZ80のもつ強力な命令であるといえます。図II-8にブロック転送命令グループを示します。

これらのブロック転送命令を使う前に次の3つのペアレジスタに転送パラメータを設定しておく必要があります。

HL：転送するデータの先頭アドレス

DE：転送先の先頭アドレス

BC：転送バイト数

LDI(LoaD Increment)命令は，HLで指定されるアドレスの内容をDEで指定されるアドレスへ1バイトだけコピーし，その後HL, DEの内容をインクリメント(＋1)し，BCは

図II-7　交換命令グループ

			インプライドアドレッシング					
		X / ニモニック	AF′	BC′, DE′ & HL′	HL	IX	IY	動作
インプライド	AF	EX AF, X	08					A ↔ A′ F ↔ F′
	BC DE HL	EXX		D9				B ↔ B′ C ↔ C′ D ↔ D′ E ↔ E′ H ↔ H′ L ↔ L′
	DE	EX DE, X			EB			D ↔ H E ↔ L
レジスタ間接	(SP)	EX (SP), X			E3	DD E3	FD E3	(SP) (SP+1) ↔ HL, IX, IY

デクリメント(－1)するというものです。

LDIR(LoaD Increment and Repeat)命令はLDI命令の拡張でBCレジスタがゼロになるまでデータの転送を繰り返し一括して行います。

LDD(LoaD Decrement)とLDDR命令は上のふたつとよく似ていますが，データ転送後HL，DEがデクリメントされる点が異なります。

このブロック転送命令の大きな特徴はブロック間に重なりがあってもよいことです。ただこのとき，ソースとデスティネーションの相対位置に応じてLDIR，LDDRのどちらを使うか考慮する必要があります。

　ソースの最小アドレス<デスティネーションの最小アドレス→LDDR
　ソースの最小アドレス>デスティネーションの最小アドレス→LDIR

どうしてこうなるか考えてみてください。

ブロックサーチ命令

ブロックサーチ命令は連続したメモリ領域の中の特定の1バイトデータを探し出すのに使用します。図II-9にブロックサーチ命令グループを示します。

これらのブロックサーチ命令を使う前に次の3つのレジスタにサーチパラメータを設定する必要があります。

図II-8　ブロック転送命令グループ

			ソース レジスタ間接 (HL)	ニモニック	動　作
デスティネーション	レジスタ間接	(DE)	ED A0	LDI	(DE)←(HL)，HL＝HL＋1，DE＝DE＋1，BC＝BC－1
			ED B0	LDIR	(DE)←(HL)，HL＝HL＋1，DE＝DE＋1，BC＝BC－1 BC＝0まで繰り返す
			ED A8	LDD	(DE)←(HL)，HL＝HL－1，DE＝DE－1，BC＝BC－1
			ED B8	LDDR	(DE)←(HL)，HL＝HL－1，DE＝DE－1，BC＝BC－1 BC＝0まで繰り返す

図II-9　ブロックサーチ命令グループ

		サーチロケーション (HL)	ニモニック	動　作
照合レジスタ	A	ED A1	CPI	A－(HL)　HL＝HL＋1，BC＝BC－1
		ED B1	CPIR	A－(HL)　HL＝HL＋1，BC＝BC－1 BC＝0またはA＝(HL)まで繰り返す
		ED A9	CPD	A－(HL)　HL＝HL－1，BC＝BC－1
		ED B9	CPDR	A－(HL)　HL＝HL－1，BC＝BC－1 BC＝0またはA＝(HL)まで繰り返す

HL：サーチするアドレス
BC：サーチするバイト数
A　：サーチするデータ

CPI(ComPare Increment)命令はHLレジスタで指定されるメモリの内容とAレジスタのデータを1回だけ比較しHLを+1, BCを－1するものです。その結果フラグレジスタのZフラグは一致した場合のみ1になります。

CPIR命令はCPI命令の拡張でBCレジスタがゼロになるまで比較を繰り返します。ただし，目的のデータ(Aレジスタと同じデータ)がみつかれば(Zフラグ＝1)命令は自動的に終了します。HLレジスタは一致したデータがあるアドレスの次のアドレスを示すので，その時点のHLレジスタの値よりデータの格納位置がわかります。

CPDとCPDR命令は上のふたつと同様な命令ですが，比較の後HLレジスタをデクリメントする点が異なり，逆方向のメモリサーチ時に使用します。

算術と論理演算命令

図II-10に8ビット算術演算命令グループを示します。これらのうちINC, DEC命令以外はAレジスタとソースデータとの間の操作が行われ，結果は比較命令(CP)を除いて，Aレ

column 3

Z-80のアドレス表現の注意点

Z80CPUではアドレスや16ビットの数字を表す2バイトの数字オペランドはメモリ格納時はすべて下位バイト，上位バイトの順にしなければなりません。たとえばメモリの1234番地の内容をBCレジスタにコピーするという命令，

LD BC,1234H

のニモニックを3000Hアドレスからマシン語に変換すると，次のようになります。

ロード命令のニモニック

この他のコール（CALL）命令あるいはジャンプ（JP）命令におけるアドレスの指定も同じです。ただし，アセンブラを使用する場合はこれを意識する必要がなく，ソースプログラム（ニモニック）では普通どおりに入力するとアセンブラが自動的に下位バイト，上位バイトの順にメモリに格納してくれます。

ジスタにおかれます。CP 命令では実行の前後でAレジスタの内容は変化しません。またフラグレジスタはこれらの操作の結果に応じて変化します。命令によるフラグの変化は後にまとめて説明します。

図II-11 にAレジスタあるいはキャリーフラグを操作する命令グループを示します。

図II-12 に 16 ビット演算命令グループを示します。

ローテイトとシフト命令

ローテイト，シフト命令はレジスタやメモリの内容をビット単位で左または右に回転またはずらす命令で，かけ算，割り算を行う場合によく使われます。

ローテイトとシフト命令グループを図II-13 に示します。

図II-10　8ビット算術，論理演算命令グループ

ソース

	レジスタ・アドレッシング							レジスタ間接	インデックスド		イミディエット	
ニモニック ＼ X	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n	動　作
ADD A, X	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n	8ビット加算 A←A+ソース
ADC A, X	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n	キャリー付8ビット加算 A←A+ソース+Cy
SUB X	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n	8ビット引き算 A←A－ソース
SBC A, X	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n	キャリー付8ビット引き算 A←A－ソース－Cy
AND X	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n	論理積 A←A∧ソース
XOR X	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n	排他的論理和 A←A⊕ソース
OR X	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n	論理和 A←A∨ソース
CP X	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n	比較 Aの内容は不変，フラグだけが変化する
INC X	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d		8ビットインクリメント ソース←ソース+1
DEC X	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d		8ビットデクリメント ソース←ソース－1

図II-11　レジスタ制御命令グループ

ニモニック	マシンコード	動　　作
DAA	27	Aレジスタに対し10進補正する　　A $\xleftarrow{BCD}$ A
CPL	2F	Aレジスタの各ビットを反転(0→1，1→0)する　　A ← $\overline{A}$
NEG	ED 44	Aレジスタに対しての補数をとる　　A←0－A
CCF	3F	キャリーCyを反転する　　Cy ← $\overline{Cy}$
SCF	37	キャリーCyをセットする　　Cy ← 1

この中でRRDとRLD命令は少し他と異なり，HLレジスタペアで指定されるメモリ内の2桁(4ビットで1桁を表す)とAレジスタ内の1桁とのローテイトを行う命令で，BCD

図II-12　16ビット演算命令グループ

ソース

ニモニック ＼ X	BC	DE	HL	SP	IX	IY	動　作
ADD HL, X	09	19	29	39			16ビット加算　HL←HL＋ソース
ADD IX, X	DD 09	DD 19		DD 39	DD 29		16ビット加算　IX←IX＋ソース
ADD IY, X	FD 09	FD 19		FD 39		FD 29	16ビット加算　IY←IY＋ソース
ADC HL, X	ED 4A	ED 5A	ED 6A	ED 7A			キャリー付16ビット加算　HL←HL＋ソース＋Cy
SBC HL, X	ED 42	ED 52	ED 62	ED 72			キャリー付16ビット減算　HL←HL－ソース－Cy
INC	03	13	23	33	DD 23	FD 23	16ビットインクリメント　ソース←ソース＋1
DEC	0B	1B	2B	3B	DD 2B	FD 2B	16ビットデクリメント　ソース←ソース－1

図II-13　ローテイト/シフト命令グループ

ソースおよびデスティネーション

ニモニック ＼ X	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	動　作
RLC X	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06	ローテイトレフト サーキュラ
RRC X	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E	ローテイトライト サーキュラ
RL X	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16	ローテイト レフト
RR X	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E	ローテイト ライト
SLA X	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26	シフト レフト アリスメティック
SRA X	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E	シフト ライト アリスメティック
SRL X	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E	シフト ライト ロジカル
RLD								ED 6F			ローテイト レフト デシマル
RRD								ED 67			ローテイト ライト デシマル

ニモニック	マシン語	動　作
RLCA	07	RLC A と同じ
RRCA	0F	RRC A と同じ
RLA	17	RL　A と同じ
RRA	1F	RR　A と同じ

ータの演算に利用できます。

ビット操作命令

ビット操作命令は汎用レジスタやメモリアドレス内の任意のビットを0か1に設定，あるいは0か1の判定する機能を持ち，パソコンによる周辺機器の制御によく使われます。

図II-14　ビット操作命令グループ

	レジスタ・アドレッシング							レジスタ間接	インデックスド		
ニモニック ＼ X	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	動　作
BIT 0, X	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46	ビット判定 "0" → Zフラグセット ("1") "1" → Zフラグリセット ("0")
BIT 1, X	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E	
BIT 2, X	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56	
BIT 3, X	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E	
BIT 4, X	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66	
BIT 5, X	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E	
BIT 6, X	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76	
BIT 7, X	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E	
RES 0, X	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86	ビットリセット
RES 1, X	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E	
RES 2, X	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96	
RES 3, X	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E	
RES 4, X	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6	
RES 5, X	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE	
RES 6, X	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6	
RES 7, X	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE	
SET 0, X	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6	ビットセット
SET 1, X	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE	
SET 2, X	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6	
SET 3, X	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE	
SET 4, X	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6	
SET 5, X	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE	
SET 6, X	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6	
SET 7, X	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE	

図II-14 にビット操作命令グループを示します。

ビット(BIT) 命令は指定するレジスタまたはメモリのビットが 0 ならばFレジスタのZフラグをセット(1)し，1ならばZフラグをリセット(0)します。

リセット(RES)命令は指定するレジスタまたはメモリのビットをリセット(0)します。

セット(SET)命令は RES 命令と反対に指定するレジスタまたはメモリのビットをセット(1)します。

ジャンプ，コール，リターン命令

ジャンプ命令はプログラムの流れを変えるためのものであり，Z80CPU では各種の条件に対応するジャンプ命令が用意されています。一方コールとリターン命令は，組み合わせて使う命令で BASIC 言語の GOSUB～RETURN と同様の働きをします。ジャンプ命令と同様に Z80 には条件付コール命令がいくつかあります。

ジャンプ命令はいくつかのアドレッシングモードでジャンプ先のアドレスの指定ができます。図II-15 にジャンプ，コール，リターン命令グループを示します。

この図の DJNZ 命令は，Bレジスタと組み合わせて使用し，相対ジャンプのループ制御を行う命令です。この命令が実行されるごとにBレジスタがデクリメントされ，0でなければ DJNZ のオペランド e に指定した相対アドレスへのジャンプ(ループ)を繰り返し，0ならば次のステップに移ります。

図II-16 にコール命令の一種であるリスタート命令グループを示します。これらの命令はマシン語1バイトでそれぞれ図に示した特定のアドレスへコールを行います。

図II-15　ジャンプ，コール，リターン命令グループ

X ／ ニモニック	無条件	C キャリ	NC ノン・キャリ	Z ゼロ	NZ ノン・ゼロ	PE 偶数パリティ	PO 奇数パリティ	M 負	P 正	カウント	動作
JP X, nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n		絶対アドレスジャンプ
JR X, e	18 e	38 e	30 e	28 e	20 e						相対アドレスジャンプ
JP (HL)	E9										
JP (IX)	DD E9										レジスタ間接ジャンプ
JP (IY)	FD E9										
CALL X, nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n		コール
DJNZ e										10 e	Bレジスタによる相対ジャンプループ
RET X	C9	D8	D0	C8	C0	E8	E0	F8	F0		リターン
RETI	ED 4D										マスク可能な割り込みからのリターン
RETN	ED 45										マスク不可能な割り込み (NMI) からのリターン

入力，出力命令

入力，出力命令はCPU汎用レジスタあるいはメモリと外部入出力デバイスとのデータ転送を実行します。外部入出力デバイスＩ/Ｏポートの指定は直接方式(ｎ)とレジスタ間接方式(ｃ)があり，(ｎ)の場合256のＩ/Ｏポート番号で指定できるのに対し，(ｃ)の場合は基本的には(ｎ)と同じですがＡレジスタ以外のレジスタへの入力ができる点が異なります。またＢレジスタと組み合わせて行うと64KバイトのＩ/Ｏポートの指定ができます。X1では主として後者の方式で64KバイトのＩ/Ｏ空間を制御できるようになっています。これについては次の章で詳しく説明します。

また256バイトまでのブロック単位の入出力を行う命令もあり前述したブロック転送，サーチ命令に似た働きをしますがレジスタの設定は異なります。ブロック入出力命令はレジスタを次のように設定してから行います。

Ｂ：入出力バイト数

Ｃ：Ｉ/Ｏポートのアドレス(番号)

図II-16　リスタート命令グループ

ニモニック	マシン語	動作		
RST 00H	C 7	リスタートアドレス	0000H	に対するコール
RST 08H	C F		0008H	
RST 10H	D 7		0010H	
RST 18H	D F		0018H	
RST 20H	E 7		0020H	
RST 28H	E F		0028H	
RST 30H	F 7		0030H	
RST 38H	F F		0038H	

図II-17　入力命令グループ

ニモニック ＼ X	A	B	C	D	E	H	L
IN X, (n)	DB n						
IN X, (C)	ED 78	ED 40	ED 48	ED 50	ED 58	ED 60	ED 68

ブロック入力命令グループ

		ソース(入力ポート) レジスタ間接 (C)	ニモニック	動作
レジスタ間接	(HL)	E D A 2	INI	インプット インクリメント (HL)←(C), HL←HL+1, B←B−1
		E D B 2	INIR	インプット インクリメント リピート (HL)←(C), HL←HL+1, B←B−1, B=0まで繰り返す
		E D A A	IND	インプット デクリメント (HL)←(C), HL←HL−1, B=B−1
		E D B A	INDR	インプット デクリメント リピート (HL)←(C), HL←HL−1, B=B−1, B=0まで繰り返す

HL：入力(出力)データ格納先頭アドレス

図II-17，図II-18 に入出力命令グループを示します。

CPU制御命令

図II-19 に 7 つのCPU 制御命令を示します。このうちとくに DI,EI,IM0～2 は割り込み制御に関する命令です。X1 で使用されている割り込みモード 2 (IM=2) については第III章のサブ CPU の節で説明します。

図II-20 に各種の命令によって F (フラグ) レジスタの各フラグビットがどのように変化するかをまとめて示します。表に出ていないその他の命令はフラグに影響を与えません。

この表はマシン語プログラミングの際，命令の実行結果を調べて次の動作をさせるときなどにそれぞれの命令のフラグ変化をみるのに使います。

図II-18　出力命令グループ

1バイト出力命令グループ

ニモニック ＼ X	A	B	C	D	E	H	L
OUT (n), X	D3 n						
OUT (n), X	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69

ブロック出力命令グループ

			ソース レジスタ間接 (HL)	ニモニック	動作
出力ポート	レジスタ間接	(C)	E D A 3	OUTI	アウト インクリメント (C)←(HL)，HL←HL+1，B=B−1
			E D B 3	OTIR	アウト インクリメント リピート (C)←(HL)，HL←HL+1，B=B−1，B=0まで繰り返す
			E D A B	OUTD	アウト デクリメント (C)←(HL)，HL←HL−1，B=B−1
			E D B B	OTDR	アウト デクリメント リピート (C)←(HL)，HL←HL−1，B=B−1，B=0まで繰り返す

図II-19　CPU制御命令グループ

ニモニック	マシン語	動作
NOP	00	何もしない命令，時間かせぎ等に使うこともある
HALT	76	CPUの動作を次に割り込みが入るまで停止させる
DI	F3	マスカブル割り込みによる割り込みを禁止する
EI	FB	マスカブル割り込みによる割り込みを可能にする
IM 0	ED 46	それぞれ割り込みモード0，1，2を設定する
IM 1	ED 56	
IM 2	ED 5E	

図II-20　フラグ変化表

命　　令	C	Z	P/V	S	N	H	備　　考
ADD A, s; ADC A, s	↕	↕	V	↕	0	↕	8ビット加算，キャリーを含む加算
SUB s; SBC A, s, CP s	↕	↕	V	↕	1	↕	8ビット減算，キャリーを含む減算，比較
NEG	↕	↕	V	↕	1	↕	符号反転
AND s	0	↕	P	↕	0	1	論理演算
OR s; XOR s	0	↕	P	↕	0	0	
INC	●	↕	V	↕	0	↕	8ビット・インクリメント
DEC s	●	↕	V	↕	1	↕	8ビット・デクリメント
ADD dd, ss	↕	●	●	●	0	X	16ビット加算
ADC HL, ss	↕	↕	V	↕	0	X	16ビットキャリーを含む加算
SBC HL, ss	↕	↕	V	↕	1	X	16ビットキャリーを含む減算
RLA; RLCA; RRA; RRCA	↕	●	●	●	0	0	ローテイト・アキュムレータ
RL s;RLC s;RR s;RRC s SLA s; SRA s; SRL s	↕	↕	P	↕	0	0	ローテイト，シフトS
RLD, RRD	●	↕	P	↕	0	0	ローテイト・デジット　左，右
DAA	↕	↕	P	↕	●	↕	デシマル・アジャスト・アキュムレータ
CPL	●	●	●	●	1	1	アキュムレータ補数変換
SCF	1	●	●	●	0	0	セット・キャリー
CCF	↕	●	●	●	0	X	キャリー補数変換
IN r, (C)	●	↕	P	↕	0	0	レジスタ間接入力
INI; IND; OUTI; OUTD	●	↕	X	X	1	X	ブロック入出力
INIR; INDR; OTIR; OTDR	●	1	X	X	1	X	BC≠0ならばZ＝0，その他はZ＝1
LDI, LDD	●	X	↕	X	0	0	ブロック転送
LDIR, LDDR	●	X	0	X	0	0	BC≠0ならばP/V＝1，その他はP/V＝0
CPI, CPIR, CPD, CPDR	●	↕	↕	X	1	X	ブロック・サーチ A＝(HL)ならばZ＝1，その他はZ＝0 BC≠0ならP/V＝1，その他はP/V＝0
LD A, I; LD A, R	●	↕	IFF	↕	0	0	IFFの内容がP/Vにコピーされる
BIT b, s	●	↕	X	X	0	1	Sのビットbの内容がZにコピーされる

記号の説明

記号	意味	説明
C	キャリー/リンク・フラグ	結果のMSBからのキャリーがあれば，C＝1
Z	ゼロ・フラグ	0ならば，Z＝1
S	サイン・フラグ	結果のMSBが1ならば，S＝1
P/V	パリティとオーバフロー兼用フラグ	結果が奇数，またはオーバフローならば，P/V＝1 結果が偶数ならば，P/V＝0
H	ハーフ・キャリー	結果にキャリー，ボローがあれば，H＝1
N	加算/減算フラグ	さきの演算が減算ならば，N＝1
↕		操作の結果，変化する
●		操作の結果，変化しない
0		操作により，リセットされる
1		操作により，セットされる
X		無視してよい
V		オーバーフラグとして扱われる
P		パリティフラグとして扱われる
s		8ビットロケーション
R		リフレッシュカウンタ
I		Iレジスタ（割り込みベクトルの上位バイト用）
r		CPUレジスタ　A，B，C，D，E，H，L
ss		16ビット・ロケーション
n		8ビット値（0〜255）
nn		16ビット値（0〜65535）
b		1ビット値（0〜7）

アセンブリ言語について

マシン語プログラムを作る場合，前にも述べたように直接にマシン語を使うよりもアセンブリ言語およびアセンブラを使う方がずっと簡単に作成できます。アセンブリ言語のメリットは単に複雑な16進のマシン語コードをニモニックで表現できる便利さだけではありません。ニモニックをマシン語に変換していく過程で生ずるメモリアドレスのめんどうな計算，たとえばサブルーチンコールやジャンプ時のアドレス指定などをラベルを使用することによってアセンブラが自動的に行ってくれるため，プログラマの負担を大幅に軽減することもメリットのひとつです。それではアセンブリ言語について少し詳しく説明していきます。

アセンブリ言語で書かれたプログラムをソースプログラムと呼びます。ソースプログラムはステートメントの集合からなります。一般に1ステートメントは1行に書き，次に述べる1ないし4つの要素からなります。

ラベル

オペコード

オペランド

コメント（これは必ずしも必要ありません）

例として次のステートメントをあげます。

```
LOOP1: LD     A,B     ;BT
ラベル オペコード オペランド コメント
```

ステートメントの要素

ラベルおよびコメント

ラベルは16ビットまでの値を示す文字列で，アドレスまたはデータに代えて用います。プログラムの処理内容が理解しやすいような文字列を付けるとよいでしょう。文字列の長さとしてはアセンブラにもよりますが，たとえば初めの6文字が有効でそれ以上は無視される，最初の文字は英文字でなければならないというものがあります。またラベルであることを示すためにラベル名の後尾にコロン『：』を付けなければなりません。

セミコロン『；』につづいて書かれる文字はコメントとして扱われます。コメントはプログラムの見やすさ保守のしやすさのために，ソースプログラムのメイン部分などに記入しておく説明であり，マシン語コードへの変換の対象にはなりません。

ラベルやコメントは全部のステートメントにつける必要はなく，必要な場合だけに付けます。

表現式

アセンブリ言語のマシン語に対する利点として，ソースプログラムのステートメント中に表現式(演算式)を使用することができます。たとえば，定数値の値を持つMAXDATの

3倍の値をBCレジスタにロードするステートメントは次のように書くことができます。

LD　BC, MAXDAT ＊ 3

Z80アセンブリ言語に含まれるこのような演算機能はアセンブラによって多少の違いはありますが，一般的に次のような演算機能が含まれます。

演算	機能
＋	加算
－	減算
＊	乗算
／	除算
AND	論理積
OR	論理和

数値定数

アセンブリ言語は次のような文字を数値定数の後につけることによって各進数の数字を扱うことができます。

B……2進数

D……10進数

O……8進数

H……16進数

たとえば，

1111B＝15D＝17O＝0FH

10進数の場合，Dを省略することができます。また16進数で，数値定数の先頭がA～Fの英文字の場合はラベルと区別するためにその前に0をつける必要があります。つまりA123Hは0A123Hとするといった工合です。

文字列

文字列は『"』で囲んで表します。

"Oh! MZ"のように使います。

擬似命令

擬似命令はアセンブラに対してアセンブリ作業に指示を与えるだけで，命令そのものがZ80のマシン語に直接変換されるわけではありません。ただDB命令などでは命令のオペランドがマシン語に変換されるものがあります。

アセンブラの擬似命令を次に示します。

ORG nnnn：以後のアセンブルアドレスをnnnnHからに設定する。

EQU nnnn：この行のラベル名の値をnnnnとし，プログラム内でそのラベル名がオペランドとして現れるところにその値nnnnを設定する。

END ：ソースプログラムの終わりを示し，この命令をソースプログラムの最後につけないとアセンブリ作業は正しく行えません。

DB nn ：アセンブル時この行の位置(アドレス)に値nn(1バイト)を設定する。

DB "sss…"：アセンブル時この行の位置(アドレス)以降に文字列sss…のASCIIコードを1バイトずつ順次設定する。

DW nnnn：アセンブル時この行の位置(アドレス)と次のアドレスに2バイトの値nnnnを下位バイト，上位バイトの順に設定する。

DS nn ：アセンブル時この行の位置(アドレス)からnnバイト分だけメモリ領域を確保する。この命令はワークエリアなどを確保するときに使います。

本書のアセンブルリストの見方

図II-21にアセンブラの出力の一例を示します。ソースプログラムをアセンブルしてできるこの出力をアセンブルリストと呼びます。右側はプログラマがエディタなどによって入力したソースプログラムと同じリストであり，左側はアセンブラによって生成されたオブジェクトプログラムリストです。実行アドレスとソースプログラムのニモニックに対応するマシン語命令で構成されます。一般にアセンブラによってこのようなアセンブルリストファイルと同時に目的のマシン語ファイルも同時に生成されますが，アセンブルリストから直接マシン語を入力したい場合はアセンブルリストの左側のアドレスからマシン語コードを入力すればよいのです。

図II-21　アセンブラの出力例

```
                             ASEG
                             ORG     0C000H
                   TRNS49    EQU     0B54H
                   INTTAB    EQU     0054H

C000  21 1A C0               LD      HL,BEEP1
C003  22 54 00               LD      (INTTAB),HL        ;INT.ADRRESS SET
C006  06 07                  LD      B,07H
C008  11 13 C0               LD      DE,INTTIM
C00B  1A           ISETLP:   LD      A,(DE)
C00C  CD 54 0B               CALL    TRNS49
C00F  13                     INC     DE
C010  10 F9                  DJNZ    ISETLP
C012  C9                     RET
                   ;
C013  D0 81 54 15  INTTIM:   DB      0D0H,81H,54H,15H   ;INT.TIMER DATA
C017  10 0F 00               DB      10H,0FH,00H
                   ;
C01A  F5           BEEP1:    PUSH    AF
C01B  C5                     PUSH    BC
C01C  D5                     PUSH    DE
C01D  E5                     PUSH    HL
C01E  CD F7 07               CALL    07F7H              ;IOCS BEEP ROUTINE
C021  E1                     POP     HL
C022  D1                     POP     DE
C023  C1                     POP     BC
C024  F1                     POP     AF
C025  C9                     RET
                   ;
                   END
```

マシン語の一番簡単な入力方法はBASICを起動したあと，モニタに入って直接キー入力する方法です。本書に出てくるサンプルプログラムも動作の確認の段階ならその方法で十分といえます。

X1用アセンブラはすでにいろいろ発表されています。たとえば「Oh! MZ」(1985年1月号)に紹介されている「EDASM」はコンパクトで使いやすいものです。ディスクを内蔵あるいは接続していて，本格的に開発しようと思われる方はやはりCP/Mを購入されるとよいと思います。

なおプリンタ出力の印字で数字の「0」とアルファベットの「O」は，間違いやすいので注意してください。

メモリ空間とI/O空間

メモリ空間とI/O空間の考え方

■メモリとメモリ空間

コンピュータの基本的なハードウェアは① CPU(中央演算処理装置)，②メモリ(記憶装置)，③ I/O(入出力制御装置)の3つのユニットから成りたっています(図II-22)。

CPUの働きは，入力端子群(データバス)に信号の組み合わせ(命令)が入力されると，それに応じた処理を行うことです。

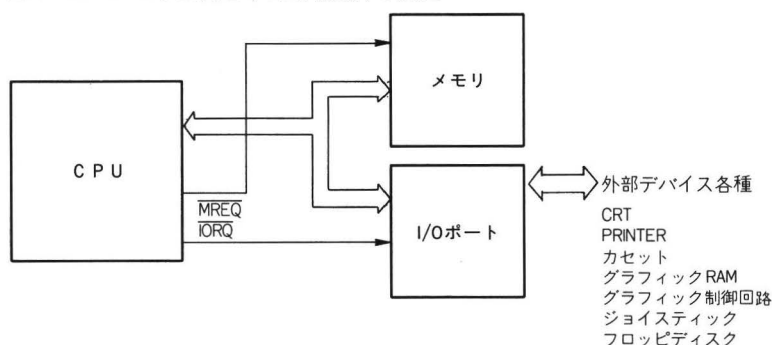
CPUの命令は分類してみるとたいして複雑ではないのですが，細かく見てみると多種多様です。ただしひとつの命令で実行する処理はごく限られたものですので，ある仕事をさせるにはそれに応じた命令を適宜与える必要があります。

しかし，ひとつずつ命令を与えていたのでは電卓と同じですし，大変な作業になってしまいます。このためコンピュータの実行の前にあらかじめプログラム(命令の並び)をメモリにしまっておくことが必要となります。

メモリには，命令を記憶するのとまったく同じようにCPUが処理するデータも記憶することができます。ですからメモリの内容の一部をひと目見ただけでは，CPUが実行する命令なのか，プログラム中で扱うデータなのかはわかりにくいものです。

メモリユニットを構成するメモリチップ(記憶素子)は大きく次のふたつに分けることができます。

1. ROM(Read Only Memory)
2. RAM(Random Access Memory)

図II-22　メモリ空間とI/O空間選択の仕組

ユーザーが作ったプログラムを入れるのは，当然書き込み可能なRAMです。

0あるいは1という情報を保持するメカニズムの実現方法から分類するとRAMはさらに，

①スタティックRAM

②ダイナミックRAM

に分けることができます。

スタティックRAMは，アクセスが速くメモリ回路の構成は簡単ですが，消費電力が大きく，発熱に十分考慮する必要があります。

これに対しダイナミックRAMは，消費電力が小さく，発熱が少なく，集積度が上げられます。ただしダイナミックRAMは，放っておくとすぐに内容が失われてしまうので，たえず記憶内容の読み出し再書き込み(リフレッシュ)をしなければなりません。最近のパソコンは記憶容量が大きくなっているので，主メモリは普通ダイナミックRAMで構成します。

8ビットCPUであるZ80は，メモリのひとつひとつの最小単位を指定するアドレス信号線を16本持っています。したがって，$2^{16}=65536$パターンの組み合わせでメモリの場所(つまり番地)を指定できます。あるひとつの番地にはデータ線8本(1バイト)のデータが格納できるので，最大65536バイトの容量のメモリを指定することができます。

コンピュータの世界では，数字はゼロから数えるので，8ビットCPU系では0番地〜65535番地，16進数で書くと0000H〜FFFFH番地になります。1Kバイト$=2^{10}=1024$ですから64Kバイト量のメモリを持つわけです。

CPUによってアクセスできるメモリ領域のことをメモリ空間といいます。Z80では特別な手段(使用しないメモリを電気的に一時切り離すなど)を使わない一般の場合では，最大64Kバイトのメモリ空間があることになります。

昔—コンピュータの世界では10年はおろか5年前さえも“昔”を意味します—は64Kバイトもあれば十分という感じでしたが，今はそうはいかずいろいろ苦労しています。その点16ビットのCPUはアドレスバスの本数がはるかに多いためとても便利です。

■I/OとI/O空間

メモリに記憶された命令やデータをCPUに入力し，その出力信号をまたメモリに記憶させるという機能だけでは，コンピュータにはなりません。外部への操作，計算結果などを知るためには何らかの出力装置に表示あるいは格納しなければなりません，また外部からの操作が必要な場合もあります。

入出力(I/O)インタフェースは，CPUと外部周辺機器とのデータ授受を受け持つところです。キーボードからCPUにデータを入力し，CPUからCRTやプリンタにデータを表示したり出力したり，カセットやディスクなど外部記憶装置とデータのやり取りを行います。

このようにCPUと外部機器との中間にあって，データの受け取り，受け渡しの管理をしているので，入出力インタフェースのことを狭い意味でI/Oポート(港)とも呼びます。

I/Oポートを通して各種外部機器が選択されますが，I/Oポートの選択もメモリと同様に，アドレス信号の組み合わせによって行います。選択することのできるI/O領域をI/O空間といいます。

CPUがメモリ空間をアクセスするかI/O空間をアクセスするかの区別は，CPUに与える命令で出力されるコントロール信号が異なるため，この信号を利用して行います。

Z80Aのメモリの1234H番地の内容を，CPU内のAレジスタにロード(コピー)するという命令についてみてみましょう。

```
LD      A,(1234H)
```

と命令されますが，CPUがこのコマンドを受け付けると，アドレスに1234H信号を出して，メモリの番地を指定します。そして，メモリ空間を呼び出すことを表すメモリリクエスト信号($\overline{\text{MREQ}}$)を有効状態(0)にしてから，書き込みでなく読み出しであることを表すリード信号($\overline{\text{RD}}$)を有効状態(0)にします(信号名の上のバーは0のときに有効，つまり負論理を示しています)。

メモリユニットは，これに応じて1234H番地のメモリの内容をデータバスに出します。するとCPUはそのデータをAレジスタに取り込みます。

一方I/Oのポート1234HからCPUのAレジスタにデータを読み込むには，

```
LD      BC,1234H
IN      A,(C)
```

を実行します。CPUにIN命令を与えると，CPUはアドレスバスに1234H信号を出し，I/O番地を指定します。I/Oリクエスト($\overline{\text{IORQ}}$)信号を有効状態にしてから，リード信号($\overline{\text{RD}}$)を有効状態にし，データが1234Hポートからデータバスにのると，CPUがそれをAレジスタに取り込みます。

以上の例からわかるように，CPUに与える命令によって，$\overline{\text{MREQ}}$信号や$\overline{\text{IORQ}}$信号が有効状態になるので，それを利用してメモリ回路やI/O回路を設計すればよいことがわかります。

メモリ空間の構成

■X1のメモリ

メモリに関するX1の仕様(図II-23)をまとめてみました。

X1 turboでは，X1/C/DではBASICに含まれているIOCS部をさらに強化してBIOS ROMとして内蔵したため，ROMは大幅に容量が増えました。

IPL(BIOS)ROMと主メモリはメモリ空間上にあります。メモリ空間は8ビットCPUでは64Kバイトだけであるはずなのに，IPL(BIOS)ROMと主メモリの合計64Kバイトを超えています。不思議に思われるでしょう，これはいわゆるバンク切り換えという方法で実現しているのです。

キャラクタジェネレータROMはCPUと直接の関係は持たず，CRTコントローラの管

理下にあってCRT上に表示する文字フォントデータを記憶しているROMです。

テキスト用V-RAM，グラフィック用V-RAMはI/O空間上にあります。ユーザー定

図II-23　X I のメモリ構成仕様

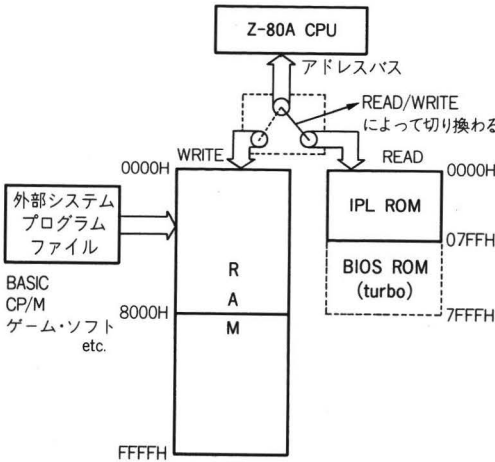
	メモリ構成	X I	XI turbo
ROM	IPL (BIOS) ROM	4KB	32KB
	CG（キャラクタジェネレータ）ROM	2KB	8KB
	漢字ROM	（オプションの基板）	128KB(第1種)
RAM	主メモリ	64KB	64KB
	テキスト用V-RAM	4KB	6KB
	グラフィック用V-RAM	48KB	96KB
	PCG（ユーザー定義CG）	6KB	6KB

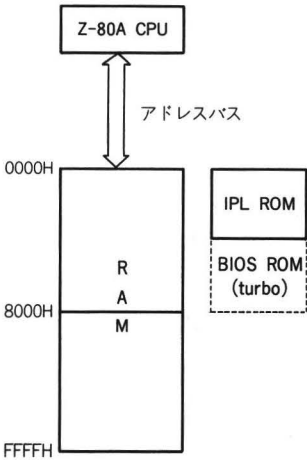
義キャラクタジェネレータ用RAMは，特殊な接続がされています。

メインメモリの64Kバイトは，64Kビットのダイナミック RAM チップ8個で構成されています。あるアドレスを指定すると並列にすべてのチップに伝わり，それぞれから対応するデータ(0か1)が出力され，それぞれのチップに対応したデータバス中の1本ずつにデータが出力されます。書き込みも同様です，ひとつのチップから1バイト入出力されないところに注意しましょう。

■IPLの動作

IPL(BIOS)ROMはメモリ空間上にありますが，これは電源投入時とバンク切り換えを行ったときです(図II-24)。IPL ROMは，メモリ空間の0000H～0FFFHにあり(X1 turbo

図II-24　電源投入時IPL起動時の
メモリマップ

図II-25　システムプログラム実行時の
メモリマップ

Z-80A CPU
アドレスバス
0000H
IPL ROM
R
A
M
BIOS ROM
(turbo)
8000H
FFFFH

のBIOSROM中のIPL部も同じ場所)，電源投入後CPUがリセットされると，このIPL(BIOS)ROMが選択されるように設計されていてIPLプログラムが0000H番地から実行されます。

具体的には，時刻やテレビタイマーの設定，さらにフロッピディスク，BASIC ROMカード，カセットの接続状態を検出し，それらのデバイスからシステムプログラムファイル(BASIC, CP/M, マシン語モニタ，ゲームプログラムなど)を64Kバイトのプログラム用RAMの0000H番地以降のRAMエリアへイニシャルローディングを実行します。

IPLプログラム実行中には，ROMとRAMのエリアが0000H〜7FFFHで重なっています。読み出し動作(CPUのリード信号)でIPL ROMの内容を読み出し，書き込み動作(CPUのライト信号)でRAMに書き込みを行うように設計されています。これによってIPL ROMとメインメモリを区別しているのです。8000H〜FFFFHのRAMエリアにおいては$\overline{\mathrm{RD}}$(リード信号)も$\overline{\mathrm{WR}}$(ライト信号)もアクセスできるようになっています。

IPL ROMは，外部システムプログラムのイニシャルローディング後，メモリ空間上から切り離され，メモリ空間は64Kバイトのプログラム用RAMの構成に切り換わります(図II-25)。ただしタイマー設定時(BASICのASKステートメントなど)には再起動されます。

X1は，前述したように電源投入時に，IPL ROMのIPLプログラムによって，システムソフトウェアが外部記憶装置からRAM上に読み込まれます。すなわち外部からのシステムソフトウェア読み込み用プログラムが，IPLの中に入っているのです。

外部記憶装置としてフロッピディスクROM，カセットテープがサポートされています。

電源投入後，IPLが起動しハードとソフトの初期設定が行われます。続いてキーが押されているかどうかを調べます。キー内容とデバイスの対照は図II-26のとおりです。

図II-26　キー入力とデバイスの対応

キー内容	選択デバイス
Fまたはfまたはハ	フロッピディスク
Rまたはrまたはス	ROM
Cまたはcまたはソ	カセット

キーが押されていれば，IPLはそのデバイスの読み込みルーチンにジャンプし接続状態を調べ，条件が満たされていれば読み込みを開始します。接続条件が満たされないときや何らかのエラーが起きたときには，再びデバイス選択キー入力待ち状態になります。

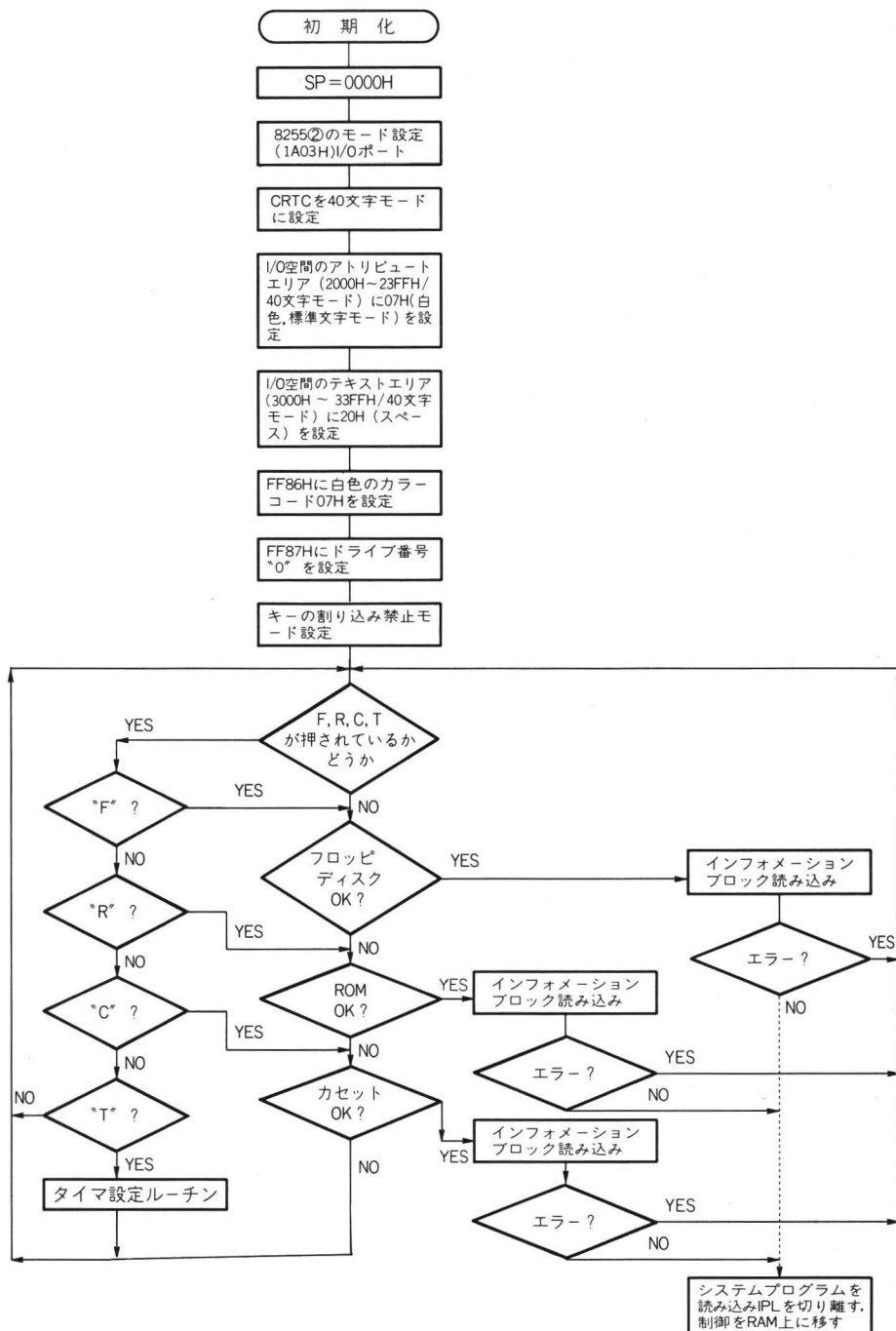
このとき，『T』または『t』，『カ』が押されていれば，IPL内のタイマー設定ルーチンが起動されます。キーが何も押されていないときは，各デバイスの接続状態を，次の優先順位で調べていきます。

① フロッピディスク，② ROM，③ カセット

接続条件とは，フロッピディスクなら電源が入っていてしかもインタフェースが正しく装備されていることです。ROMの場合は拡張I/OポートにROMボードが差し込まれていることです。カセットは最後に選択されます。

選択されたデバイス内のファイルから，システムソフトウェアの情報を持つインフォメ

ーションブロックを，メモリ上にロードします。続いてインフォメーションブロックの先頭のモード部分をみて，マシン語プログラムファイルであれば，インフォメーションの情報にしたがってシステムプログラムをメモリ上にロードし，終了後IPL ROMを切り離してメモリ上のシステムプログラムに制御を移します(図II-27)。

図II-27　IPL動作フローチャート

この間に，何らかのエラーが発生すればIPLはデバイス選択キー入力待ち状態に戻ります。

X1 turboでは，IPL起動時に初期モードスイッチの内容を読み込んで接続されているCRTの種類(解像度)とフロッピディスクの種類を知ります。

スイッチはI/Oポート1FF＊H(＊は0〜Fのどれでもよいことを示す)に設定してあるのでプログラムで読むには，

```
LD        BC,1FF0H
IN        A,(C)
```

とします。

■インフォメーションブロックの構成

インフォメーションブロックは，デバイスの種類によらず共通のフォーマットになっており，32バイトで構成され9つの部分に分けることができます(図II-28)。詳細は次のとおりです。

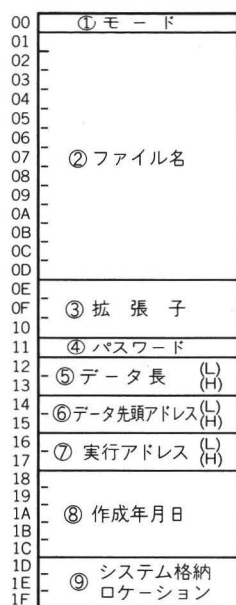
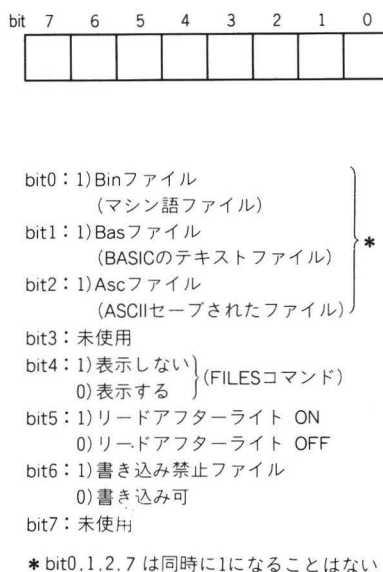
図II-28　インフォメーションブロックの構成

バイト	内容
00	①モード
01〜0D	②ファイル名
0E〜10	③拡張子
11	④パスワード
12〜13	⑤データ長 (L)(H)
14〜15	⑥データ先頭アドレス (L)(H)
16〜17	⑦実行アドレス (L)(H)
18〜1C	⑧作成年月日
1D〜1F	⑨システム格納ロケーション

図II-29　ファイル構成

bit	7	6	5	4	3	2	1	0

bit0：1)Binファイル(マシン語ファイル)
bit1：1)Basファイル(BASICのテキストファイル)
bit2：1)Ascファイル(ASCIIセーブされたファイル)　＊
bit3：未使用
bit4：1)表示しない　0)表示する　(FILESコマンド)
bit5：1)リードアフターライト ON　0)リードアフターライト OFF
bit6：1)書き込み禁止ファイル　0)書き込み可
bit7：未使用

＊bit0，1，2，7は同時に1になることはない

① **モード**　(バイト00H)

ファイルの種類を表します(図II-29)。

a. フロッピディスクの場合

00　：KILLされたファイル

FF：ディレクトリテーブルの終わり

b. ROM，カセットの場合

bit0〜3まではフロッピディスクと同様ですがbit3〜7は未使用となっています。

このバイトはIPLロード時に01H，つまりマシン語プログラムファイルでなければなりません。

② **ファイル名** （バイト01H～0DH）

13文字までのファイル名を与えます。スペースの場合は20Hです。

③ **拡張子** （バイト0EH～10H）

3文字の拡張子エリア。フロッピディスクでIPLロードファイルの場合，必ずSysでなければなりません。

④ **パスワード** （バイト11H）

無指定の場合20Hを入れてください。Hu-BASICでファイルをセーブするときパスワードをつけるには，ファイル名のあとに『；』をはさんで入力します。TEST.BASというファイル名を『I』というパスワードをつけてセーブしたい場合は，

```
SAVE "TEST.BAS;I"
```

というように入力すれば，パスワードをつけることができます。

⑤ **データ長** （バイト12H～13H）

ファイルの長さを示し12Hは下位，13Hは上位バイトを示します。

たとえば2000Hバイト長のファイルならば，図II-30のようになります。この部分はマシン語ファイルとBASICのテキストファイルのときだけ有効です。

図II-30 データ長

バイト	12H	13H
	00	20

⑥ **データ先頭アドレス** （バイト14H～15H）

ファイルロード時のメモリ先頭アドレス格納エリア(図II-31)です。ただしマシン語ファイルだけ有効です。

図II-31 データ先頭アドレス

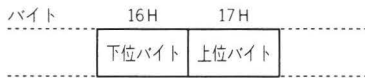
バイト	14H	15H
	下位バイト	上位バイト

⑦ **実行アドレス** （バイト16H～17H）

ロードされたプログラムの実行開始アドレスを，メモリ上のアドレスで指定します。

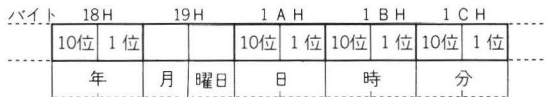
マシン語ファイルだけ有効です(図II-32)。

図II-32　実行アドレス

バイト	16H	17H
	下位バイト	上位バイト

⑧ **作成年 月曜 日 日；時分**（バイト18H～1CH）

ファイルの作成年月日および時間の格納エリア。格納フォーマットは図II-33のとおりです。

図II-33　ファイル作成年月日時分・格納フォーマット

バイト	18H		19H		1AH		1BH		1CH	
	10位	1位			10位	1位	10位	1位	10位	1位
	年		月	曜日	日		時		分	

年，日，時，分はBCD記法で格納し，月，曜日は2進数で格納します(図II-34)。

図II-34　月，曜日と数値の対応

月	1	2	3	4	5	6	7	8	9	10	11	12
数値	1	2	3	4	5	6	7	8	9	A	B	C

曜日	日	月	火	水	木	金	土
数値	0	1	2	3	4	5	6

たとえば図II-35の例は，

図II-35　月；曜日と数値の対応具体例

18H	19H	1AH	1BH	1CH
82	C1	27	17	58

82年12月(月)27日17時58分です。

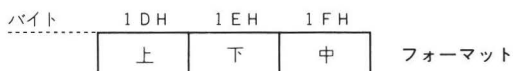
⑨ **システム格納ロケーション**（バイト1DH～1FH）

外部デバイス上のどの位置からファイル本体が格納されているかを示します。デバイスによって次のように異なります。

a. フロッピディスクの場合

ファイル本体が格納されているレコード番号を示します(図II-36)。

図II-36　ファイル本体格納レコード

バイト	1DH	1EH	1FH	
	上	下	中	フォーマット

たとえば図II-37の例は，

図II-37　ファイル本体格納レコード具体例

バイト	1DH	1EH	1FH
	00	20	00

なら，ファイルが000020Hレコード目から格納されていることを意味します。

b. ROMの場合

ファイルが格納されているROM上のアドレスを示します(図II-38)。

図II-38　ファイル格納・ROMアドレス

バイト	1DH	1EH	1FH	
	下	中	上	フォーマット

たとえば図II-39の例は，

図II-39　ファイル格納・ROMアドレス具体例

バイト	1DH	1EH	1FH
	10	0F	00

なら，ファイルが0F10Hアドレスから格納されていることを意味します。

c. カセットテープの場合

常に00が格納されています(図II-40)。カセットテープの場合，ファイルがシリアルに格納されており，ソフトで完全に制御できるため，この部分が不要になります。

図II-40　カセットファイル格納レコード

バイト	1DH	1EH	1FH	
	00	00	00	フォーマット

インフォメーションブロックは，フロッピディスクの場合，第0レコード(0面，0トラック，1セクタ)にあり，最初の32バイトで構成されます。

ROMの場合は，最初の32バイトがインフォメーションブロックになります。

カセットの場合は，シリアルな記憶媒体なので，システムファイル本体の前にインフォメーションブロックを作っておけば，テープのどこにあってもかまいません。

リストII-2

X1CP/M　V2.2のインフォメーションブロック　(0レコード)

```
#Device=0:Record no.= 0
#Adr. =      HEX DATA                                   'Character code
#00000=01 58 31 20 43 50 2F 4D 20 56 32 2E 32 20 53 79 ' X1 CP/M V2.2 Sy
#00010=73 20 00 2B 00 D4 00 EA 83 27 07 11 55 00 01 00 's  + ﾔ ｧｲ'  U
```

BASIC　CZ8FB01のインフォメーションブロック　(0レコード)

```
#Device=0:Record no.= 0
#Adr. =      HEX DATA                                   'Character code
#00000=01 42 41 53 49 43 20 43 5A 38 46 42 30 31 53 79 ' BASIC CZ8FB01Sy
#00010=73 20 00 A8 00 00 00 00 82 C1 27 17 58 00 20 00 's  ｨ    ﾁ' X
```

■IPL(BIOS)ROMのアクセス

電源を入れてIPLがシステムプログラムを読み込んだあと，IPLが切り離されます。しかし，場合によってIPL ROMを再びアクセスしたいことがあります。

たとえばX1の特徴であるタイマーを設定したいときや外部デバイスのファイル読み込みルーチンなどのルーチンを使用したいときです。X1 turboの場合はとくに有用なサブルーチン群が32Kバイトも BIOS ROM に入っているのでなおさらです。いったいどうすればいいでしょうか。

メモリ空間から切り離された状態を，ノンアクティブ(non active)状態といいます。反対に IPL ROM がアクティブ状態にあるときは，前に説明したように，7FFFH アドレス以下のメモリに対する書き込みは RAM に，読み出しは IPL ROM に対して行われます。

8000H アドレス以上のメモリに対しては，RAM しかないので，書き込みも読み出しも，当然 RAM に対して行われます。

IPL状態設定

IPL(BIOS)ROMをアクティブ状態にする

このように，電源投入時とタイマー設定時以外 IPL ROM はノンアクティブ状態にあるため，IPL ROM をアクセスするときはアクティブ状態にしてからでなければなりません。

それにはシステム I/O (1D ＊＊) Hポートを使用します(下位バイトはダミー)。このポートに対して出力命令を実行すればよいのです(出力命令ならどれでもいい)。

```
LD      BC,1D00H
            下位バイトはどんな値でもかまわない
OUT     (C),A → Aの内容はダミー。レジスタもAでなく
                他のレジスタ（B，C，D，E，H，L）のどれでもよい
```

IPL(BIOS)ROMをノンアクティブ状態にする

IPL ROM にもう用がなくなり，メモリ空間からそれを切り離すときは，システム I/O (1E ＊＊) Hポートを使用します。このポートに対して出力命令を実行すればよいのです。このときもアクティブにするときと同じく，出力命令ならどれでもかまいません。

```
LD      BC,1E00H
            下位バイトはどんな値でもかまわない
OUT     (C),A
```

■アクセス例

IPLプログラムをみるには

X1用システムプログラムを作るためには，IPL プログラムをよく理解することが必要です。まず IPL プログラムを目前に持ってこなければなりません。そうすれば BASIC 内蔵のモニタで扱うことも楽になります。簡単にそれをする方法は，IPL プログラムを IPL ROM から RAM 上に転送することだと思います。リストII-3にそのプログラムを紹介します。

このプログラムを，Hu-BASIC のモニタや CP/M の 管理下でメモリに書き込んで実行すれば，DE レジスタに設定するアドレスから4K バイトのIPL プログラムが格納されます。

もちろん正常に戻ってこられるように，格納アドレス領域を選ぶ必要があります。すな

わち，BASICやCP/Mの本体，ワークエリア以外のエリアを選ぶことです。

リストII-3
IPLプログラムをRAMに転送

```
LD      BC,1D00H          ]
OUT     (C),A             ] IPLアクティブ
LD      HL,0000H          ]
LD      DE,START          ;START=転送先先頭アドレス
LD      BC,1000H
LDIR
LD      BC,1E00H          ]
OUT     (C),A             ] IPLノンアクティブ
RET                       ]
```

タイマ設定ルーチンコールプログラム

IPLプログラムを応用する一例として，タイマー設定ルーチンをコールするプログラム(リストII-4)を紹介します。

column 4

Z80CPUの入出力命令について

Z80CPUの入出力命令は2グループに分けられます。

① Cレジスタを使用した入出力命令グループ

```
OUT     (C),r   ]
IN      r,(C)   ] Cレジスタを使用した入出力命令グループ
```

② Aレジスタを使用し，直接ポート番号を指定する入出力命令

```
IN      A,(n)   ]
OUT     (n),A   ] Aレジスタを使用し，直接ポート番号を指定する入出力命令グループ
```

第1グループは実行すると命令実行前のレジスタBの内容がアドレスバスの上位側（AD_8～AD_{15}）に出力され，レジスタCの内容がアドレスバスの下位側（AD_0～AD_7）に出力されます。

一方，第2グループは実行すると命令実行前のレジスタAの内容がアドレスバス上位側（AD_8～AD_{15}）に出力され，オペランドnの値がアドレスバスの下位側（AD_0～AD_7）に出力されます。

したがってX1のバンクの切り換えのように実際の出力と関係なくアクセスするだけならば，次のふたつのどちらでもよいことになります。

```
①   LD      BC,1234H  ]
     OUT     (C),r     ] 5バイト

②   LD      A,12H     ]
     OUT     (34H),A   ] 4バイト
```

たとえば，X1の場合，IPLROMをactiveまたは，nonactive状態にするにはそれぞれ（1D00)Hと（1E00)Hポートに対して単に何かひとつの出力命令を執行することによって行え，データバスは全く関係ありませんので上のどちらの方法を用いてもできることがわかります。長さからみると第2の方法の方が1バイト少なくてすみます。

もちろん，データバスが関係する場合には上の方法ではできません。

リストII-4

タイマー設定ルーチンコールプログラム

```
KVECOO  EQU     0133H
KVECIN  EQU     012DH
TIMER   EQU     0003H
ACCPRT  EQU     0013H

        DI                  :割り込み禁止
        LD      L,00H       ]キー割り込み禁止
        CALL    KVECOO      ]
        LD      BC,1D00H    ]IPLアクティブ
        OUT     (C),A       ]
        CALL    TIMER       :タイマー設定ルーチン
        LD      BC,1E00H    ]IPLノンアクティブ
        OUT     (C),A       ]
        CALL    KVECIN      :キー割り込みベクトル設定
        EI                  :割り込み許可
        LD      A,0CH       ]画面クリア
        CALL    ACCPRT      ]
        RET                 :リターン
```

このプログラムはHu-BASICのASKステートメントと同様な働きをします。実行する前にはタイマー設定ルーチンの性格上，X1の表示モードを40文字表示，SCREEN0,0モードに設定する必要があります。ESCキーでリターンします。

またこのプログラムはHu-BASIC内のIOCSの中のいくつかのルーチンを使用しています。したがって，X1 turboでは動かない(アドレスが違う)ので注意してください。

実行するといつものタイマー設定ルーチンが起動され，タイマーを設定できます。リターンするにはESCキーを押します。WIDTH80にして実行するとどうなるか試してみてください。動くことは動きますが……。

BOOTするには

BOOTというのは，システムを再起動させることをいいます。ふつうは，あるシステムプログラムの管理下を離れてIPLに戻るときに実行します。

BOOTするにはIPL ROMをアクティブ状態にしてから，0000Hアドレスへジャンプすればいいのです(リストII-5)。

リストII-5

```
DI                  :割り込み禁止
LD      BC,1D00H    ]IPLROMアクティブ
OUT     (C),A       ]
JP      0000H       :0000Hアドレスジャンプ
```

I/O空間の制御

■ X1のI/O構成

X1では64KバイトのI/O空間を持ち，Z80CPUの入出力命令によってアクセスされます。Z80CPUでは，Cレジスタを使用した入出力命令，すなわち，

```
IN      r,(C)  ┐
               ├ 入出力命令第1グループ
OUT     (C),r  ┘
```

を実行すると，アドレスバス上にBCレジスタの内容が出力されます。具体的には，Bレジスタの内容は上位8本のアドレスバス(AD_8～AD_{15})に，Cレジスタの内容は下位8本のアドレスバス(AD_0～AD_7)に，それぞれ出力されます。

このB，Cレジスタとインプット/アウトプット命令を合わせて使用することによって，2^{16}=64KバイトのI/Oポートの制御が実現できるわけです。

一方，Z80CPUには，インプット/アウトプット命令のほかに，

```
IN      A,(n)  ┐
               ├ 入出力命令第2グループ
OUT     (n),A  ┘
```

という入出力命令があります。

このふたつの命令は，実行すると1バイト定数nの値が下位8本のアドレスバス(AD_0～AD_7)に出力されます。nが0～255までの値の定数なので，基本的にはこのふたつのコマンドでは256のI/Oポートしか制御できません。

X1は64KバイトのI/O空間を持ちますが，X1に特有な機能として，そのI/Oアドレスを変化させることができるのです。グラフィックのスピードを上げるためにふたつのI/Oアドレスのマップを用意しました。そのモードは次のふたつです。

1．シングルアクセスモード

2．同時アクセスモード

I/O空間のアドレスマップは，それぞれのモードに応じて異なり，図II-41，42のようになっています。

シングルアクセスモードではI/Oポートを通して，ユーザーI/Oポート，システムI/Oポート，テキストおよび属性V-RAM，そしてグラフィックV-RAM1，2，3をアクセスすることができるので通常のモードといえます。

同時アクセスモードでは，I/Oポートを4つの部分に分け2画面以上のグラフィックV-RAMを同時にアクセスすることができるようになっています。具体的には図II-43のようです。

同時アクセスモードは，2画面以上のグラフィックV-RAMに，同一データを書き込むとき(画面塗りつぶしやクリアなど)の高速化を目的に設計されたモードです。

図II-43　同時アクセスモードのI/Oポートと
グラフィック画面V-RAMの対応

I/Oポート	同時アクセスグラフィック画面 V-RAM
0000H～3FFFH	BLUE, RED, GREEN
4000H～7FFFH	RED, GREEN
8000H～8FFFH	BLUE, GREEN
C000H～FFFFH	BLUE, RED

■I/Oポートのアクセス

I/Oポートをアクセスするには，前述したようにBCレジスタを使用した入出力命令によって行うことができます。

具体的にはBCレジスタにアクセスしたいI/Oポートアドレスを設定します。

```
LD        BC,****H
```

アクセスしたいI/Oポート

続いて入力なら，

```
IN        r,(C)
```

出力なら，

図II-41　X1のI/Oマップ

シングルアクセスモード

アドレス	内容
0000H	ユーザI/Oポート
1000H	システムI/Oポート
2000H	属性 V-RAM
3000H	テキスト V-RAM
3800H	
4000H	グラフィック V-RAM（青）
8000H	グラフィック V-RAM（赤）
C000H	グラフィック V-RAM（緑）
FFFFH	

同時アクセスモード

アドレス	内容
0000H	グラフィック V-RAM（赤，緑，青）
4000H	グラフィック V-RAM（赤，緑）
8000H	グラフィック V-RAM（緑，青）
C000H	グラフィック V-RAM（赤，青）
FFFFH	

```
OUT        (C),r
```

を実行することによってアクセスできます。

rは，入力ならデータを受け取るレジスタを表し，出力ならrレジスタの内容が出力されます。レジスタとしてA, B, C, D, E, H, Lレジスタが使えます。

■シングルアクセスと同時アクセスモードの設定

X1のI/Oポートの アクセスには，ふたつのモードがあり，同一ポートでもモードによってアクセス内容が違います。したがってI/Oポートをアクセスする場合には，まずアクセスモードを知ること，あるいは設定することが必要になります。

シングルアクセスモードの設定

I/Oポートのアクセスモードの設定について述べます。

モード切り替えは，システムI/Oポート(1A02H)のD_5に割り当てられています。このポートに立ち下がり信号を出力する(“1”信号を出力してから“0”信号を出力する)ことによって，シングルアクセスモードと同時アクセスモードの切り替えができるようになっています。

シングルアクセスモード時は，各I/Oポートに対して 読み出しも書き込みも可能ですが，同時アクセスモード時は書き込みしかできません。

図II-42　X1 turboのI/Oマップ

シングルアクセスモード

アドレス	バンク0		バンク1
0000H	ユーザーI/Oポート		
1000H	システムI/Oポート		
2000H	属　性 V-RAM		
2800H			
3000H	テキスト V-RAM		
3800H	漢　字 V-RAM		
4000H	グラフィック V-RAM (青)		グラフィック V-RAM (青)
8000H	グラフィック V-RAM (赤)	バンク切換 ↔ I/O 1ED0H	グラフィック V-RAM (赤)
C000H	グラフィック V-RAM (緑)		グラフィック V-RAM (緑)
FFFFH			

同時アクセスモード

アドレス			
0000H	グラフィック V-RAM (赤, 緑, 青)		グラフィック V-RAM (赤, 緑, 青)
4000H	グラフィック V-RAM (赤, 緑)	バンク切換 ↔ I/O 1FD0H	グラフィック V-RAM (赤, 緑)
8000H	グラフィック V-RAM (緑, 青)		グラフィック V-RAM (緑, 青)
C000H	グラフィック V-RAM (赤, 青)		グラフィック V-RAM (赤, 青)
FFFFH			

このことはI/Oポートに対して入力命令を実行すれば，アクセスモードがいかなる状態であっても，シングルアクセスモードに切り替わるハード構成になっているからです。

シングルアクセスモードを設定するには，単に入力命令を実行すればよいことになります。

```
IN        A,(C)
```

このとき，Cレジスタは意味を持たずどんな値でもよいのです。Aレジスタの内容は変わる可能性があるので注意が必要です。それまでのAの値を保存しておきたければ次のようにします。

```
PUSH      AF        :Aレジスタの値を退避
IN        A,(C)     :シングルアクセスモード設定
POP       AF        :Aレジスタに復帰
```

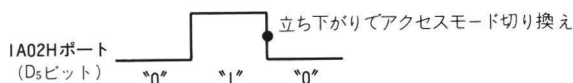
たとえば，レジスタAの内容をシングルアクセスモードのI/Oポート(3000H)に出力し，しかもAの内容を保持したいときはリストII-6のようにします。

```
リストII-6

PUSH      AF
IN        A,(C)
POP       AF
LD        BC,3000H
OUT       (C),A
.
.
.
```

同時アクセスモードの設定

同時アクセスモードを設定する手順は図II-44のようになります。

図II-44　アクセスモード切り換え制御図

```
リストII-7
IN        A,(C)      :シングルアクセスモード設定
LD        BC,1A02H   :BCレジスタにI/O(1A02H)を設定
IN        A,(C)      :I/OポートからデータをAレジスタに入力
AND       DFH        :①
OUT       (C),A      :Aレジスタの内容を(1A02H)ポートから出力 ∴D5=0
SET       5,A        :Aレジスタの第5ビットをセット(1にする)
OUT       (C),A      :Aレジスタの内容を(1A02H)ポートから出力 ∴D5=1
RES       5,A        :Aレジスタの第5ビットをリセット(0にする)
OUT       (C),A      :Aレジスタの内容を(1A02H)ポートから出力 ∴D5=0
.
.
.
① AレジスタとDFH (11011111)2 のANDをとる。第5ビットD5が0になる。他のビットは不変
```

まずシングルアクセスモードに一度したのち，I/O(1A02H)ポートの第5ビット(D_5)から“0”→“1”→“0”の順にデータを出力すればよいのです。

プログラムで書くと，リストII-7のようになります。

モード設定後，たとえばI/O (3000H) ポートへ“32H”を出力したい場合は，モード設定に続いてリストII-8のように行います。

リストII-8

```
LD      A,32H     ：出力したいデータをAレジスタに設定
LD      BC,3000H  ：BCレジスタにI/Oポート（3000H)を設定
OUT     (C),A     ：Aレジスタの内容を出力
 .
 .
 .
```

■ユーザーI/OポートとシステムI/Oポート

ユーザー用および システム用I/Oポートは，シングルアクセスモード時だけアクセスすることができます。したがってこれから述べることは，とくに断わらないかぎりシングルアクセスモードを前提とします。

ユーザー I/O ポートとは，X1のユーザーがオリジナルな周辺回路やデバイスをX1に接続して使用する場合や，メーカーが提供する周辺機器のための入出力ポートです。

I/O アドレスの 0000H～0FFFH の 4K バイトの空間を持ちます。ただしこのうち，

0100H～0FFFH

までの I/O ポートは，メーカーがこれからも提供する周辺機器用に予約されているため，ユーザーが使用する場合は，なるべく 00FFH 以下(0000H～00FFH)のポートを使用するのが無難です。

現在メーカーが発表しているオプションのI/Oポート一覧表を示します(図II-45)。

図II-45　オプション機器のI/Oポート

I/Oアドレス		内容	備考
Bレジスタ	Cレジスタ		
0B	**	増設RAMバンク切り替え	書き込み用X1 turboのみ
0D	**	外部RAMボード	0D00 → EMM0 ～ 0D09 → EMM9
0E	**	外部ROMボード	漢字ROM ～ 0E80H ～ 0E81H BASIC ROM ～ 0E00H ～ 0E03H
0F	**	フロッピディスク	0FF8H ～ 0FFFH

システムI/Oポートとは，X1の内部回路やデバイスをアクセスするのに割り当てられた入出力ポートで，I/O アドレスの，

1000H～1FFFH

の範囲を占めています。

このシステム I/O ポートを通してパレット回路，優先順位回路，CGROM と PCGRAM, CRT コントローラ, 8255 パラレル入出力コントローラ，サブ CPU80C49, PSG, IPL

ROM などがアクセスされます。

図II-46はシステムのI/Oポートの詳細です。これらの各種の制御回路やコントローラのアクセスなどは，テキスト画面やグラフィック画面の構成も含め，この後で詳しく説明します。

図II-46　システムI/Oポート

I/Oアドレス		内　容	※…turboのみ 無印…共通
Bレジスタ	Cレジスタ		
10	＊＊	パレット　青	
11	＊＊	パレット　赤	
12	＊＊	パレット　緑	
13	＊＊	プライオリティ	
14	＊＊	CGROM（＊漢字ROM）	
15	＊＊	PCG　青	
16	＊＊	PCG　赤	
17	＊＊	PCG　緑	
18	＊0 ＊1	CRTC　1800　レジスタ 1801　データ	
19	＊＊	8255 ①　1900　ポートA 1901　ポートB 1902　ポートC	
1A	＊0 ＊1 ＊2 ＊3	8255 ②　1A00　ポートA 1A01　ポートB 1A02　ポートC 1A03　コントロールレジスタ	
1B	＊＊	PSG　データ	
1C	＊＊	PSG　アドレス	
1D	＊＊	IPL (BIOS) ROM アクティブ	
1E	＊＊	IPL (BIOS) ROM ノンアクティブ	
1F	8＊	DMA	※
	90 91 92 93	SIO　1F90　チャンネルAデータ 1F91　チャンネルAコントロール 1F92　チャンネルBデータ 1F93　チャンネルBコントロール	※
	A0 A1 A2 A3	CTC　1FA0　チャンネル0 1FA1　チャンネル1 1FA2　チャンネル2 1FA3　チャンネル3	※
	B＊		
	C＊		
	D＊	画面管理	※
	E＊	黒色制御	※
	F＊	スタートポート	※

Ⅲ X1ブラックボックスを探検する—2

●

画面構成とCRTコントロール

X1の画面構成

CRTコントローラ

テキスト画面とグラフィック画面

画面の操作

サブCPUの働きとコントロール

サブCPUの機能構成

サブCPUのコントロール機能

PSGのハイテク活用法

PSGの機能とレジスタ

効果的なサウンド作り

画面構成とCRTコントロール

X1の画面構成

■画面の考え方

X1の画面は，キーボード入力で文字を文字単位で表示するテキスト画面と，ドット単位で点や曲線などを表示するグラフィック画面のふたつで構成されています(図III-1)。もちろんCRTに表示されるのは，両画面が合成したものです。

テキスト画面とグラフィック画面のふたつから成り立っているということは，ハード的にはテキストRAMとグラフィックRAMというふたつのメモリが，独立して存在していることを意味します。画面上に文字やグラフィックを表示するためには，このメモリに画面に関するデータを入れなければなりません。

X1は，画面表示用メモリV-RAMがI/O空間にあるので，プログラム上では，Z80の入出力命令であるINやOUTで画面にアクセスします。Z80の入出力命令は，大きく分けると次の2種類です。

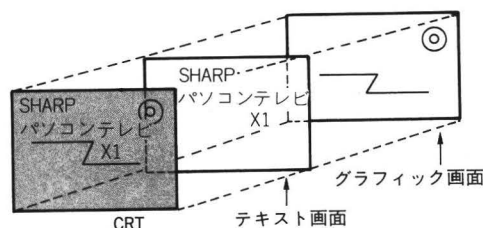
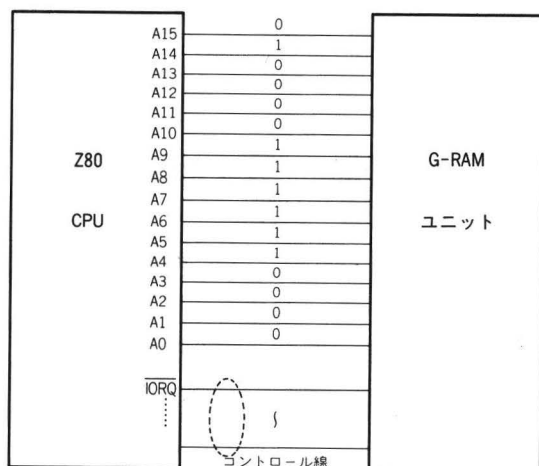
① 直接ポート番号(0～255)を指定して，Aレジスタとの間でデータをやりとりする。

② Cレジスタの値をポート番号とし，レジスタとの間でデータをやりとりする。

ここでいうポート指定は，ポート値がメモリのアドレス指定と同じように，アドレスバスに2進数で伝わることです(図III-2)。

ただし①の場合は，ポート番号が0から255(00000000～11111111(2))であるので，アドレスバスの下位8ビットまでが有効です。②の場合は，アドレスバス下位8ビットはCレジスタの内容が現れますが，実は上位8ビットにBレジスタの内容が現れています。したがってX1は，これを利用してI/O空間を大幅に広げています。このことを強調して命令を次のように書いたほうが正確といえるかもしれません。

```
IN      r,(BC)  :入力
OUT     (BC),r  :出力
```

図III-1 画面構成概念図

図III-2　I/O命令とアドレスバス

BCレジスタの値が
43F0H＝0100, 0011, 1111, 0000(2)
の場合にOUT(C), Reg. あるいは
IN Reg.,(C)の命令を実行したとき
のアドレス線の様子

通常のZ80を使ったマイコンではI/O空間が256バイトだったのをX1は64Kバイトにまで広げています(X1 I/Oマップについては図II-38を参照)。

X1の画面上には，文字を表示するテキスト画面と，図形を表示するグラフィック画面が重なって表示されると述べました。このことはメッセージなどの表示はテキスト画面にデータを書き込み，グラフィックの表示はグラフィック画面にデータを書き込めばよいという意味です。

BASICでは前者はPRINT命令に，後者はPSET, LINE, CIRCLE命令などに相当します。

■画面のモード

X1の画面は，大きく分けると，ふたつの状態(モード)をとることができます(図III-3)。

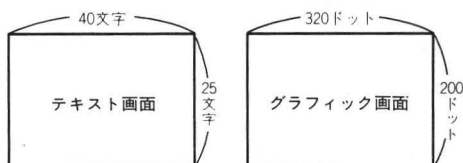
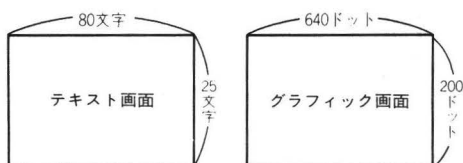
ドットとは，画面上の最小単位の1点のことです。

ⓑのモードは，キャラクタでもグラフィックでも，ⓐのモードに比べて左右方向において2倍細かくなっています。キャラクタ(テキスト)画面の1文字は，画面で8×8ドット(水平方向8ドット，垂直方向8ドット)で表示されます。

40×25文字テキスト画面モードを指定すると，グラフィック画面は自動的に320×200ドットグラフィック画面モードになり，80×25文字テキスト画面モードのときは，グラフィック画面モードは640×200ドットモードになります。

文字画面は細かい方の80×25キャラクタモードを使用し，同時にグラフィック画面を

図III-3　画面表示モード

ⓐ40×25キャラクタ／320×200ドット

ⓑ80×25キャラクタ／640×200ドット

粗いほうの320×200ドットモードで使うことはできないので注意しましょう。

色の指定については，ⓐ，ⓑモードともテキスト画面の文字は，文字単位（8×8ドット）で，グラフィック画面はドット単位で8色指定できます。

このふたつの画面状態の切り換えはBASICではWIDTH命令を用います。

ⓐ　40×25キャラクタ　320×200ドット
　　WIDTH 80　↓　　↑　WIDTH40
ⓑ　80×25キャラクタ　640×200ドット

ⓐのモードは，ⓑのモードに比べて画面が粗いという欠点があるかわりに，画面をふたつ持てるという長所があります。ディスプレイに表示されている画面のほかに，もう1枚分画面があるということです（図III-4）。

実際にディスプレイ上に表示する画面（ページ）と，書き込む画面を異なるようにすることもできます。したがって表示されないページにグラフィックスを作成し，完成したところで表示ページを変えるようにすると，一瞬のうちに次の画面を表示することも可能です。

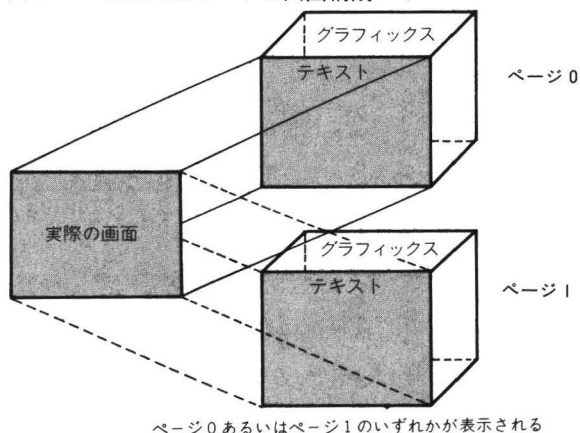
入力ページ（書き込み用）と出力ページ（表示用）の切り換えは，BASICではSCREEN（GRAPHでも同様）命令を使います。

SCREEN　x，y
　　　x；出力ページ（0または1）
　　　y；入力ページ（0または1）

X1 turboは，グラフィックV-RAM倍増の96Kバイト内蔵により従来の画面モードに加えて垂直方向400ラインの表示ができる640×400ドットのグラフィック画面を実現しているうえ，漢字やアンダーライン表示のためのモードが増えています。

さらに400ライン高解像度ディスプレイだけでなく200ラインのディスプレイにも自動的に対応できるようにするため複雑になっています（図III-5）。

これらのモードのうち，40(80)×12行，320(640)×192ドット表示モードは，低解像度モード（200ラインディスプレイ）でも漢字を表示できるようにするために用意されたものです。アンダーラインが表示できるのは，テキスト画面が10行または20行のときだけです。

図III-4　WIDTH40モードの画面構成

図III-5　X1 turboのグラフィック画面

①200ライン表示ディスプレイ用

テキスト画面	グラフィック画面	テキスト画面	グラフィック画面
40×25行	320×200ドット	80×25行	640×200ドット
40×12行	320×192ドット	80×12行	640×192ドット
40×20行		80×20行	
40×10行		80×10行	

②400ライン表示ディスプレイ用

テキスト画面	グラフィック画面	テキスト画面	グラフィック画面
40×25行	320×200ドット	80×25行	640×200ドット
40×12行	320×192ドット	80×12行	640×192ドット
40×20行		80×20行	
40×25行	320×400ドット	80×25行	640×400ドット
40×12行	320×384ドット	80×12行	640×384ドット

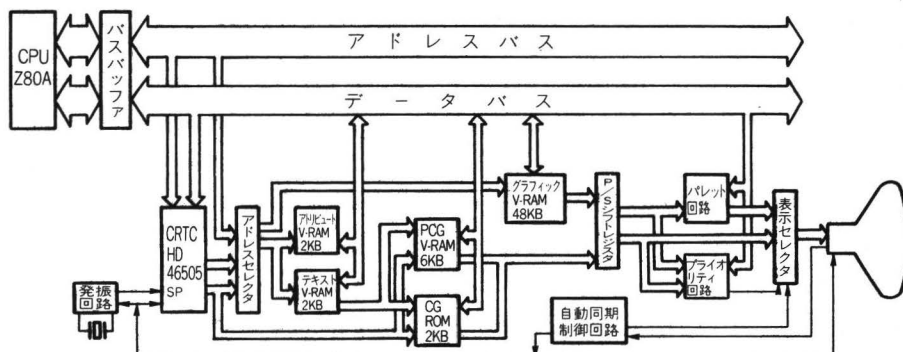
CRTコントローラ

■CRTの機能

X1の画面の制御のためにHD-46505SPという1チップCRTコントローラ(CRTC)が使われています。

このCRTCは，他のICと比べてシンプルな構造をしていて，いろいろ応用することができるという特色をもっています。パラメータによって，豊富な機能も設定できます。

グラフィック回路部分のダイアグラムを図III-6に示します。X1 turboでは，図III-6よりさらに大幅に回路が強化されています。グラフィックV-RAMはバンク分けにより，2倍の容量になり，また漢字表示のための漢字ROMが付け加えられ，さらにパレット回路，プライオリティ回路などが専用IC化されました。従来のX1シリーズでは別売であったデ

図III-6　X1グラフィック回路ダイアグラム

ジタルテロッパも内蔵しました。CRTC はテキスト V-RAM だけでなく，グラフィック V-RAM のアドレス制御も行っています。

■文字画面の表示

CRTC の働きについて，文字を画面に出す場合を例にして簡単に説明しましょう。

文字を画面に出すには，テキスト V-RAM のある番地にその文字に対応する ASCII コード(オーナーズマニュアルに表掲載)を書き込まなければなりません。V-RAM は I/O空間にあるので，書き込む命令はアセンブリ言語の LD(ロード)ではなく OUT です。

V-RAM の番地は画面上の位置に対応します。詳しくは後述しますが，画面左上隅が一番小さい番地に対応しています。その行が右にいくにつれて順にひとつずつ番地が増えます。1行が終わるとその下の行の左端にいき，というぐあいに続きそれぞれ V-RAM と対応しています。

画面上のいろいろな文字に対応した ASCII コードが V-RAM に入ると，CRTC によって自動的にディスプレイにその文字パターンが表示されます。

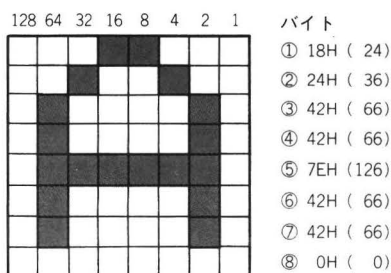
CRTC は，テキスト V-RAM のアドレスを順に発生します。そのアドレスに入っている ASCII データが，CGROM または PCGRAM に入力されると，そのコードに対応した文字パターンが出力されます。パターンのデータは，1文字につき8バイトです。

たとえば『A』というパターンは，CGROM 内に図III-7のような8バイトデータで構成されています。

CRTC はこの8バイトのデータを1バイトずつ読み出し，各バイトのビットデータにしたがって“0”ドットは“暗”を，“1”ドットは“明”を表示します。

1文字を表示するには，CRTC は8回の掃引を行います。しかし同じ行の文字を1文字ずつ8回掃引するのではなく，1回の掃引は同じ行の同一ラスタ文字に対して行われ8回のラスタ掃引で1行中の文字が表示されます。

1文字の8バイトの出力タイミングは CRTCがコントロールし，CRTCからCGROM または PCGRAM につながるバスによって行います。

図III-7 “A”文字パターン図

具体的には CGROM(PCGRAM)から出力された1ドット行ごとのバイトデータは，シフトレジスタによって，ビットごとにドット単位の直列信号に変換されます。CRTC によって作られた水平方向と垂直方向の同期信号とともに，このドット単位のデータはパレット回路，プライオリティ回路の制御を受けて TV ディスプレイで画面に表示されます。

データは青赤緑のうちの3つの基本色でそれぞれ別に出力されます。それぞれ0か1の2値信号なので，結局8色のカラーグラフィックスが実現されます。

CRTC-HD46505SP の内部レジスタの機能(図III-8)をかかげ，この図中の記号と画面構成との対応も図示します。

図III-8　CRTコントローラHD-46505SPの内部レジスタ

レジスタ番号	設定パラメータ	機　　　　能
R0	水平総文字数	水平走査の同期の指定
R1	水平表示文字数	1行当たりの表示文字数の指定
R2	水平同期位置	水平同期信号の出力位置の指定
R3	同期パルス幅	下位4ビットで水平同期信号のパルス幅を指定 上位4ビットで垂直同期信号のパルス幅を指定
R4	垂直総文字数	垂直走査の同期の指定
R5	総ラスタ調整	1フィールドの最後に付加するラスタ数の指定
R6	垂直表示文字数	画面に表示する文字行数の指定
R7	垂直同期位置	垂直同期信号の出力位置の指定
R8	インターレースとスキュー	ラスタスキャンモードおよびDISPTMG信号CUDISP信号のスキューの指定＊
R9	最大ラスタアドレス	行間スペース含む1行のラスタ数の指定
R10 R11	カーソルスタートラスタ カーソルエンドラスタ	カーソルの形状と表示モードの指定 （未使用）
R12 R13	スタートアドレス (H) (L)	リフレッシュメモリの読み出し先頭アドレスの指定
R14 R15	カーソルアドレス (H) (L)	カーソルの表示アドレスの指定 （未使用）
R16 R17	ライトペン (H) (L)	ライトペンの検出アドレスの格納 （未使用）

■画面モードの設定

CRTCレジスタ番号の設定はI/Oアドレスの1800H番地，データの設定はI/Oアドレスの1801H番地に割り当てられています。そこで，たとえばレジスタ1に28Hを設定するには，リストIII-1のようにします。

```
リストIII-1
LD        BC,1800H
LD        A,01H
OUT       (C),A
INC       BC
LD        A,28H
OUT       (C),A
```

40文字モードと80文字モードのレジスタ設定値を図III-9に示しプログラム(リストIII-2,3)をかかげます。

X1 turboとそれ以外のX1シリーズの各画面モードのCRT設定値を図III-10に示します。具体的な画面モードの詳細は後述します。

```
リストIII-2
40文字モード設定

SET40:  LD      DE,000EH                      :D＝レジスタ番号
        LD      HL,CRTCD1                     :E＝設定レジスタ数
CRT1:   LD      BC,1800H ┐
        OUT     (C),D    ┘レジスタ番号指定
        INC     BC       ┐
        LD      A,(HL)   │データ設定
        OUT     (C),A    ┘
        INC     HL
        INC     D
        DEC     E
        JR      NZ,CRT1
        LD      BC,1A03H ┐
        LD      A,0DH    │基本クロック切り換え
        OUT     (C),A    ┘
;
;       (EXIT)
;
CRTCD1: DB      37H
        DB      28H
        DB      2DH
        DB      34H
        DB      1FH
        DB      02H
        DB      19H
        DB      1CH
        DB      00H
        DB      07H
        DB      60H
        DB      07H
        DB      00H
        DB      00H
```

リストIII-3
80文字モード設定

```
SET80:  LD      DE,000EH        ; D=レジスタ番号
        LD      HL,CRTCD2       ; E=設定レジスタ数
CRT2:   LD      BC,1800H  ┐ レジスタ番号指定
        OUT     (C),D     ┘
        INC     BC        ┐ データ設定
        LD      A,(HL)    │
        OUT     (C),A     ┘
        INC     HL
        INC     D
        DEC     E
        JR      NZ,CRT2
        LD      BC,1A03H  ┐
        LD      A,0CH     │ 基本クロック切り換え
        OUT     (C),A     ┘
;
;       (EXIT)
;
CRTCD2: DB      6FH       ┐ 40文字モードとデータが
        DB      50H       │ 違うのは，この4つだけ
        DB      59H       │
        DB      38H       ┘
        DB      1FH
        DB      02H
        DB      19H
        DB      1CH
        DB      00H
        DB      07H
        DB      60H
        DB      07H
        DB      00H
        DB      00
```

図III-9　CRTコントローラのレジスタ設定値

レジスタ番号	40文字モード(HEX)	80文字モード(HEX)	内　　容
0	37	6F	水平総文字数
1	28	50	水平表示文字数
2	2D	59	水平同期位置
3	34	38	同期パルス幅
4	1F	1F	垂直総文字数
5	02	02	総ラスタ調整
6	19	19	垂直表示文字数
7	1C	1C	垂直同期位置
8	00	00	インタレースとスキュー
9	07	07	最大ラスタアドレス
10	60	60	カーソルスタートラスタ
11	07	07	カーソルエンドラスタ
12	00／04(＊)	00	スタートアドレス（H）
13	00	00	スタートアドレス（L）
14	00	00	カーソルアドレス（H）
15	00	00	カーソルアドレス（L）
16	00	00	ライトペン　（H）
17	00	00	ライトペン　（L）

＊ページ0表示…00
　ページ1表示…04

図III-10　CRTCのレジスタ設定値（X1 turbo）

低解像度モード

テキスト画面	40×25	80×25	40×12	80×12	40×20	80×20	40×10	80×10
グラフィック画面 / レジスタ番号	320×200	640×200	320×192	640×192				
0	37	6F	37	6F	37	6F	37	6F
1	28	50	28	50	28	50	28	50
2	2D	59	2D	59	2D	59	2D	59
3	34	38	34	38	34	38	34	38
4	1F	1F	0F	0F	18	18	0B	0B
5	02	02	02	02	08	08	12	12
6	19	19	0C	0C	14	14	0A	0A
7	1C	1C	0E	0E	16	16	0B	0B
8	00	00	00	00	00	00	00	00
9	07	07	0F	0F	09	09	13	13
10	00	00	00	00	00	00	00	00
11～17	00	00	00	00	00	00	00	00

高解像度モード

テキスト画面	40×25	80×25	40×12	80×12	40×25	80×25	40×12	80×12	40×20	80×20
グラフィック画面 / レジスタ番号	320×200	640×200	320×192	640×192	320×400	640×400	320×384	640×384		
0	35	6B	35	6B	35	6B	35	6B	35	6B
1	28	50	28	50	28	50	28	50	28	50
2	2D	59	2D	59	2D	59	2D	59	2D	59
3	84	88	84	88	84	88	84	88	84	88
4	1B	1B	0D	0D	1B	1B	0D	0D	15	15
5	00	00	00	00	00	00	00	00	08	08
6	19	19	0C	0C	19	19	0C	0C	14	14
7	1A	1A	0D	0D	1A	1A	0D	0D	15	15
8	00	00	00	00	00	00	00	00	00	00
9	0F	0F	1F	1F	0F	0F	1F	1F	13	13
10	00	00	00	00	00	00	00	00	00	00
11～17	00	00	00	00	00	00	00	00	00	00

CRTCには，画面のスクロール機能もあり，BASICでは，

SCROLL　n

{n＝－3～＋3の整数}

により実効されます。ただし，これはテレビ放送とコンピュータ画面が重なっているときにのみ有効な命令です。

スクロールの設定は図III-11のように行います。

図III-11　画面スクロール機能の制御

レジスタ5	0←2←4←6	8→A→C→………→1E
スクロール方向	上方向	下方向
速　度	速←――→遅	遅←――→速

① CRTCのレジスタ5にスクロール速度を設定する。パラメータは図III-11のとおりです。

② パラレルインタフェースIC8255②のポートC{I/Oポート1A03H}のビット4を0にする。

反対にスクロールを停止するためには，次のようにします。

① CRTCレジスタ5に02Hをセットする。

② 8255② {I/Oポート1A03H} のビット4を1にする。

それぞれのプログラム例(リストIII-4)を示しましょう。

リストIII-4

スクロール開始

```
        LD      BC,1800H  ┐
        LD      D,05H     │
        LD      E,04H     │ CRTCレジスタ5に04Hを書き込む
        OUT     (C),D     │
        INC     BC        │
        OUT     (C),E     ┘
        LD      BC,1A03H  ┐
        LD      A,08H     │ 8255②ポートCのビット4をリセットする
        OUT     (C),A     ┘
 ;
 ;      (EXIT)
 ;
```

スクロール停止

```
        LD      BC,1800H  ┐
        LD      D,05H     │
        LD      E,02H     │ CRTCレジスタ5に02Hを書き込む
        OUT     (C),D     │
        INC     BC        │
        OUT     (C),E     ┘
        LD      BC,1A03H  ┐
        LD      A,09H     │ 8255②ポートCのビット4をセットする
        OUT     (C),A     ┘
 ;
 ;      (EXIT)
 ;
```

テキスト画面とグラフィック画面

■テキスト画面

テキスト画面とは，その名の通り“text”つまり文章を表示するための専用画面で，おもに文章を扱うときに使用します。文章を表示するためテキスト画面では主としてパターンの決まった文字を使います。通常CGROMに文字のパターンを記憶させておき，表示のときはこのパターンを読み出して画面表示します。

X1シリーズでは，テキスト画面にユーザー定義文字も表示できるようにPCGRAMを

持ち合わせています。

これに対してグラフィック画面とは，ドット単位に任意のパターンを表示できる画面でグラフィック表示に主として使われます。グラフィック画面の取り扱いはテキスト画面よりめんどうで，動作に時間がかかるのが普通です。

テキスト画面に表示できるパターンは図III-12のとおりです。

図III-12　テキスト画面表示パターン

表示パターン	X 1	X1 turbo
キャラクタジェネレータ（CG）	アルファベットなど256種類，それぞれ8×8ドット1種類（2KB）	アルファベットなど256種類，それぞれ8×8ドット，8×16ドット2種類（8KB）
プログラマブルキャラクタジェネレータ(PCG)	8×8ドットのフォントで256種類，青緑赤それぞれ2KBで計6KB	8×8ドットのフォントで256種類，8×16ドットのフォントで128種類，青緑赤それぞれ2KBで計6KB
漢字ROM		第1水準漢字文字2965字とJIS非漢字文字453字(計128KB)と第2水準漢字3384字（オプション，128KB）

X1 turboの大きな特徴は，テキスト画面に漢字表示ができることです。X1 turbo以外のX1シリーズで漢字を表示するには，オプションの漢字ROMボードからパターンを読み込み，グラフィック画面に転送するためのソフトウェアが必要です。したがってX1 turboに比べて処理も手数がかかり時間もとります。

X1のテキスト画面

前述したように，決まった文字パターンでCGROMに記憶されているアルファベットなどを表示するには，画面上の位置に対応するV-RAM(テキストV-RAM)にその文字のASCIIコードを書き込むことが必要です。

ユーザー定義文字を表示するときも，同じテキストV-RAMを使います。この場合はPCGRAMに定義したユーザー定義番号を指定することになります。

画面にCGROMの文字を出すか，ユーザー定義文字を出すかの指定が各文字ごとに可能です。

このほか8色の指定，文字サイズ(標準，縦倍，横倍、縦横倍)，反転，点滅の指定と選択ができます。これらは画面上の位置に対応したI/O空間の属性V-RAMに，各文字1バイトのデータ設定で行います。

このデータの各ビットの内容を図III-13に示します。

テキスト画面とテキストV-RAM，属性V-RAMのアドレス対応図(図III-14)を示します。

図中の番地がテキストV-RAMのアドレスで，カッコの中の番地が属性V-RAMのアドレスを示しています。テキストV-RAMから1000Hマイナスした番地が属性V-RAMのアドレスとなっています。

画面左上隅に『A』という文字を黄色で出し，チカチカ点滅させるには，図III-15のようになります。

図III-13　属性V-RAMのビット内容

ビット番号	7	6	5	4	3	2	1	0
機　能	拡大		PCG	点滅	補色	色		

7	6	拡大
0	0	ノーマル
0	1	垂直2倍
1	0	水平2倍
1	1	垂直水平2倍

5	PCG
0	CG
1	PCG

4	点滅
0	ノーマル
1	点滅

3	補色
0	ノーマル
1	補色

2	1	0	色
0	0	0	黒
0	0	1	青
0	1	0	赤
0	1	1	赤紫
1	0	0	緑
1	0	1	水色
1	1	0	黄
1	1	1	白

図III-14　テキスト画面とV-RAMのアドレス対応

㋐40字×25行モード

ページ0（40×25）

3000 (2000)	3001 (2001)	……	3027 (2027)
3028 (2028)	3029 (2029)	……	304F (204F)
⋮	⋮		⋮
33C0 (23C0)	33C1 (23C1)	……	33E7 (23E7)

ページ1（40×25）

3400 (2400)	3401 (2401)	……	3427 (2427)
3428 (2428)	3429 (2429)	……	344F (244F)
⋮	⋮		⋮
37C0 (27C0)	37C1 (27C1)	……	37E7 (27E7)

㋑80字×25行モード（80×25）

3000 (2000)	3001 (2001)	……	304F (204F)
3050 (2050)	3051 (2051)	……	309F (209F)
⋮	⋮		⋮
3780 (2780)	3781 (2781)	……	37CF (27CF)

図III-15　"A"表示属性V-RAM値

リストIII-5のように属性V-RAMの値は16Hとします。

リストIII-5

```
LD      BC,3027H ┐ テキストV-RAM (3027H)へ
LD      A,41H    │
OUT     (C),A    ┘ 41Hを書き込む
LD      BC,2027H ┐
LD      A,16H    │ 属性V-RAM (2027H)へ
OUT     (C),A    ┘ 16Hを書き込む
```

I/Oポートにアクセスする命令がBASICにも用意されています。OUT命令とPOKE@命令がそれです。ただし，両者には少しばかり違いがあります。

```
POKE@   アドレス，データ〔，データ…〕｛指定できるアドレスの範囲：&H2000～&HFFFF｝
OUT     アドレス，データ｛指定できるアドレスの範囲：&H0000～&HFFFF｝
```

次のふたつのBASICのプログラム(リストIII-6)は同様の働きをします。

リストIII-6

```
10 SCREEN 0,0:WIDTH 40      ┐
20 LOCATE 10,10             │
30 CFLASH1:COLOR 4          │ ①
40 PRINT "A"                │
50 CFLASH0:COLOR 7          ┘

10 SCREEN 0,0:WIDTH 40      ┐
20 POKE@&H3000+10+40*10,65  │ ②
30 POKE@&H2000+10+40*10,20  ┘
```

X1 turboのテキスト画面

アルファベットなど，ASCIIコード表にある文字の表示は，従来のX1と同様1バイトのASCIIコードをテキストV-RAMに格納して行います。ただし，X1 turboはテキスト画面に漢字表示もできるためさらに以下の設定が必要です。ところで漢字表示のときは8ビットだけでは，全部の漢字を指定することができないため，X1 turboは漢字テキストV-RAMを内蔵しました。これをテキストV-RAMと合わせて使用すると，すべての漢字指定と表示ができるようになり，しかもアンダーライン機能などの指定も可能になりました。

X1 turboのテキスト，漢字テキスト，属性V-RAMの構成を図III-16に示し，CGROM文字，PCGRAM文字，漢字表示がどのように行われるかを説明します。

CRTCがこれらのV-RAMをハードウェアによって参照し，文字(漢字を含む)をテキスト画面に表示します。この場合，CGROM，PCGRAM，漢字ROMのどれから文字フォントパターンを読み込んで表示するかは，図III-16の①～③のデータのうち次のビットによって選択します(図III-17)。

・漢字テキストV-RAMのD7($\overline{\text{CG}}$/KANJI)
・漢字テキストV-RAMのD4($\overline{1}$/2水準)
・アトリビュートV-RAMのD5($\overline{\text{ROM}}$/RAM)

図III-16　テキスト画面の構成

①テキストV-RAM（I/O 3000H～37FFH）

D7	D6	D5	D4	D3	D2	D1	D0	
ASCIIコード 1								番地構造

CGROM, PCGRAMの場合：8ビットのASCIIコード
漢字ROMの場合　　　　：ROMアドレスの下位8ビット

②漢字テキストV-RAM（I/O 3800H～3FFFH）

D7	D6	D5	D4	D3	D2	D1	D0	
$\overline{CG}$/KANJI	$\overline{L}$/R	ULINE	$\overline{1}$/2水準	ASCIIコード2				番地構造

D0～D3：漢字ROMアドレス　上位4ビット
D4　　：漢字ROMアクセスの場合第1水準，第2水準を選択
　0＝第1水準　　1＝第2水準
　PCGRAMアクセスの場合，CG/KANJI記号，ROM，
　RAM信号とともに，PCGのアクセス方式を選択
　0＝PCGキャラクタモード　　1＝PCG外字モード
　CGROMアクセスの場合，無視される
D5　　：アンダーライン表示 ON/OFF
　0＝アンダーラインなし　　1＝アンダーライン表示
D6　　：漢字ROMアクセスの場合漢字フォントの左8×16ドット，
　右8×16ドットのアクセス選択
　0＝左半分　　1＝右半分
　CGROM，PCGRAMアクセスの場合は無視される
D7　　：CGと漢字ROMのアクセス選択
　0＝CGROM，PCGRAM　　1＝漢字ROM

③ 属性V-RAM（I/O 2000H～27FFH）

D7	D6	D5	D4	D3	D2	D1	D0	
H倍	V倍	ROM/RAM	BLINK	REV	緑	赤	青	番地構造

各ビットの意味

D0～D2：キャラクタの色の指定
D3　　：D0～D2で指定した色の補色(反転)表示
　0＝通常　　1＝反転
D4　　：キャラクタを約0.5秒周期で点滅
　0＝通常　　1＝点滅
D5　　：CGROM，漢字ROMとPCGRAMのアクセス選択
　0＝CGROM，漢字ROM　　1＝PCGRAM
D6　　：キャラクタの大きさの選択

キャラクタの色指定ビット表

ビット			色
D2	D1	D0	
0	0	0	黒
0	0	1	青
0	1	0	赤
0	1	1	赤紫
1	0	0	緑
1	0	1	水色
1	1	0	黄
1	1	1	白

キャラクタの大きさ指定ビット表

ビット		機　能
D7	D6	
0	0	ノーマル文字
0	1	垂直2倍文字
1	0	水平2倍文字
1	1	垂直，水平2倍文字

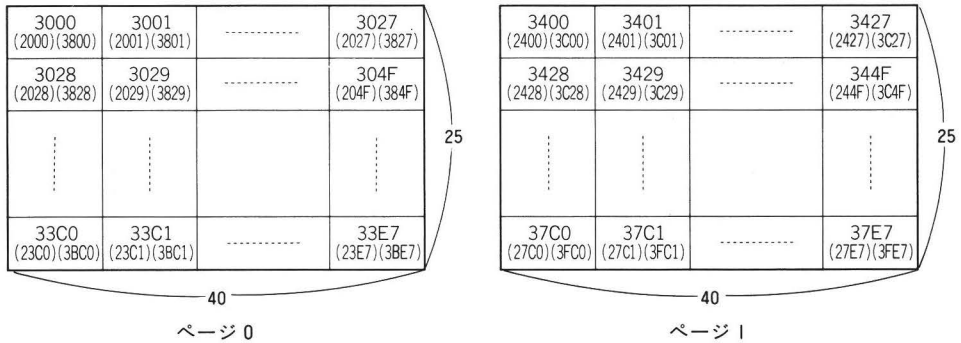
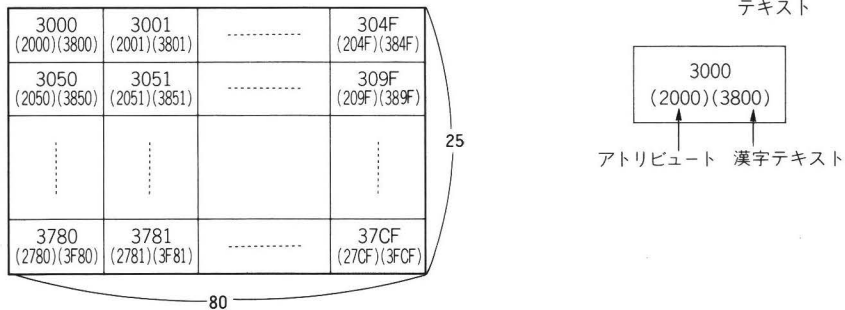
図III-17 文字フォントビットデータ

$\overline{\text{ROM}}$/RAM	$\overline{\text{CG}}$/KANJI	$\overline{\text{1}}$/2水準	表示の選択
0	0	*	CGROM
0	1	0	漢字ROM(第1水準)
0	1	1	漢字ROM(第2水準)
1	0	0	PCGRAMキャラクタ方式
1	0	1	PCGRAM外字方式

漢字フォントは，1文字を左右に2分して別のROMに格納しています。どちら側を表示するかは，漢字テキストV-RAMのD6で選択します。

画面の各モードにおける画面上の位置とV-RAMのアドレスの対応を図III-18に示します。ひとつの四角が1文字分を表しており，テキストV-RAM，アトリビュートV-RAM，漢字テキストV-RAMの各アドレスを，それぞれ四角内の上，左下カッコ内，右下カッコ内に示しています。

図III-18 テキスト画面とV-RAMのアドレス対応（X1 turbo）

㋐40字×25行モード

3000 (2000)(3800)	3001 (2001)(3801)	…………	3027 (2027)(3827)
3028 (2028)(3828)	3029 (2029)(3829)	…………	304F (204F)(384F)
⋮	⋮		⋮
33C0 (23C0)(3BC0)	33C1 (23C1)(3BC1)	…………	33E7 (23E7)(3BE7)

25　40

ページ0

ページ1

㋑80字×25行モード

3000 (2000)(3800)	3001 (2001)(3801)	…………	304F (204F)(384F)
3050 (2050)(3850)	3051 (2051)(3851)	…………	309F (209F)(389F)
⋮	⋮		⋮
3780 (2780)(3F80)	3781 (2781)(3F81)	…………	37CF (27CF)(3FCF)

25　80

㋒40字×12行モード

ページ0（40字×12行）

3000 (2000)(3800)	3001 (2001)(3801)	…………	3027 (2027)(3827)
3028 (2028)(3828)	3029 (2029)(3829)	…………	304F (204F)(384F)
⋮	⋮		⋮
31B8 (21B8)(39B8)	31B9 (21B9)(39B9)	…………	31DF (21DF)(39DF)

12
40
ページ0

ページ1（40字×12行）

3200 (2200)(3A00)	3201 (2201)(3A01)	…………	3227 (2227)(3A27)
3228 (2228)(3A28)	3229 (2229)(3A29)	…………	324F (224F)(3A4F)
⋮	⋮		⋮
33B8 (23B8)(3BB8)	33B9 (23B9)(3BB9)	…………	33DF (23DF)(3BDF)

12
40
ページ1

㋓80字×12行モード

3000 (2000)(3800)	3001 (2001)(3801)	…………	304F (204F)(384F)
3050 (2050)(3850)	3051 (2051)(3851)	…………	309F (209F)(389F)
⋮	⋮		⋮
3370 (2370)(3B70)	3371 (2371)(3B71)	…………	33BF (23BF)(3BBF)

12
80

㋔40字×20行モード

ページ0（40字×20行）

3000 (2000)(3800)	3001 (2001)(3801)	…………	3027 (2027)(3827)
3028 (2028)(3828)	3029 (2029)(3829)	…………	304F (204F)(384F)
⋮	⋮		⋮
32F8 (22F8)(3AF8)	32F9 (22F9)(3AF9)	…………	331F (231F)(3B1F)

20
40
ページ0

ページ1（40字×20行）

3400 (2400)(3C00)	3401 (2401)(3C01)	…………	3427 (2427)(3C27)
3428 (2428)(3C28)	3429 (2429)(3C29)	…………	344F (244F)(3C4F)
⋮	⋮		⋮
36F8 (26F8)(3EF8)	36F9 (26F9)(3EF9)	…………	371F (271F)(3F1F)

20
40
ページ1

㋕80字×20行モード

3000 (2000)(3800)	3001 (2001)(3801)	…………	304F (204F)(384F)
3050 (2050)(3850)	3051 (2051)(3851)	…………	309F (209F)(389F)
⋮	⋮		⋮
35F0 (25F0)(3DF0)	35F1 (25F1)(3DF1)	…………	363F (263F)(3E3F)

20
80

㋖40字×10行モード

ページ0（40×10）

3000 (2000)(3800)	3001 (2001)(3801)	…………	3027 (2027)(3827)
3028 (2028)(3828)	3029 (2029)(3829)	…………	304F (204F)(384F)
⋮	⋮		⋮
3168 (2168)(3968)	3169 (2169)(3969)	…………	318F (218F)(398F)

ページ1（40×10）

3200 (2200)(3A00)	3201 (2201)(3A01)	…………	3227 (2227)(3A27)
3228 (2228)(3A28)	3229 (2229)(3A29)	…………	324F (224F)(3A4F)
⋮	⋮		⋮
3368 (2368)(3B68)	3369 (2369)(3B69)	…………	338F (238F)(3B8F)

㋗80字×10行モード

（80×10）

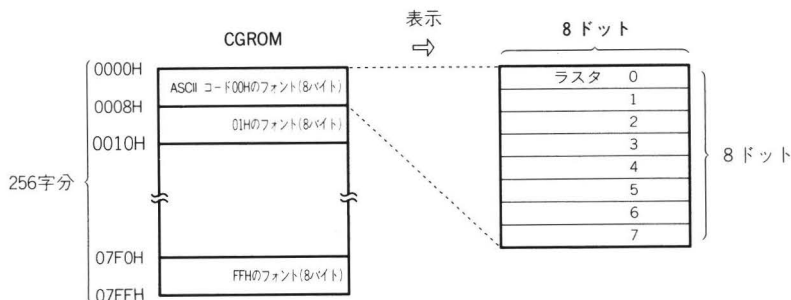
3000 (2000)(3800)	3001 (2001)(3801)	…………	304F (204F)(384F)
3050 (2050)(3850)	3051 (2051)(3851)	…………	309F (209F)(389F)
⋮	⋮		⋮
32D0 (22D0)(3AD0)	32D1 (22D1)(3AD1)	…………	331F (231F)(3B1F)

■CGROMのアクセス

CGROMフォントデータ構造

ここではCGROM文字フォントデータの構造，CGROM文字表示法，そしてCGROM文字フォントデータの読み出しについて説明します。CGROMをアクセスするには1400HのI/Oポートを使用します。

X1のCGROMは2Kバイトの容量を持ち，1文字当たり8バイト(8×8ドット)，256文字分のフォントデータが格納されています。フォントデータはCGROM中で，図III-19のように保持されています。

図III-19　X1 CGROM内文字フォントデータ構造

X1 turboではCGROMは8Kバイトの容量を持ち，次の2種類のフォントデータが格納されています。

① 8×8ドットフォントデータ(4Kバイト)

② 8×16ドットフォントデータ(4Kバイト)

8×8ドットフォントデータは低解像度モードの40(80)×25行，40(80)×20行モードのみに使われ，その他の画面モードでは8×16ドットのフォントデータが使われます。

X1 turboでは各フォントデータはCGROMの中で図III-20のように格納されています。

すなわち，前半4Kバイトに8×8ドットフォントデータが同一データを2回ずつ繰り返して格納されており，後半4Kバイトには8×16ドットフォントデータが格納されています。8×8ドットフォントデータ1文字分を読み出す場合は同一データが2バイト続けてあるため，1バイトおきに読み出すと8回の読み出しで行えます。

CGROMの文字表示

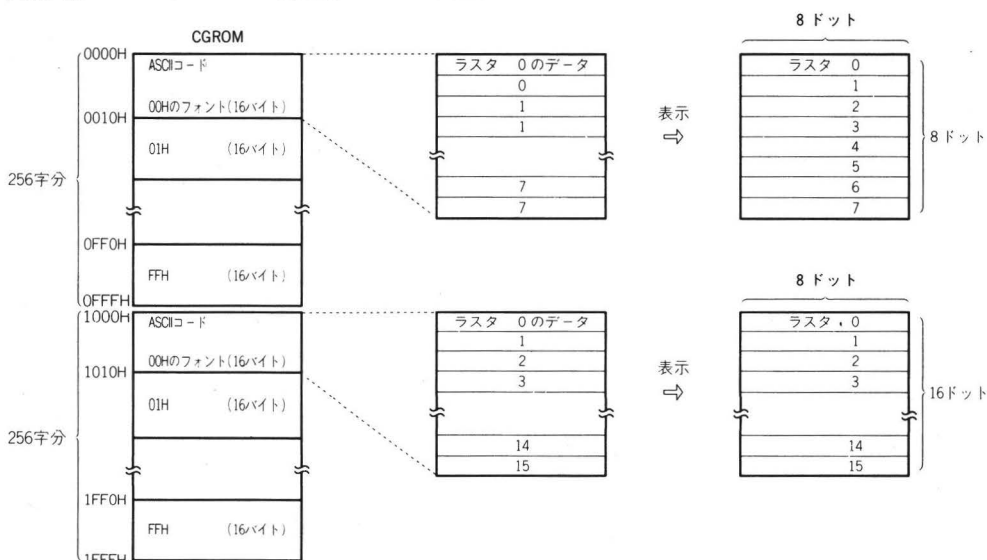
テキスト画面への表示法についてはすでに説明しましたが，ここでまとめておきます。CGROMのフォントデータを画面に表示させるためには，テキストおよびアトリビュートV-RAMを次のように設定します。

・テキストV-RAM：ASCIIコードを格納します。

・アトリビュートV-RAM：$\overline{\text{ROM}}$/RAM選択ビット(D5)を0にします。

・漢字テキストV-RAM(X1 turboのみ)：$\overline{\text{CG}}$/KANJIビット(D7)を0にします。

X1 turboでは文字を表示するときは，まえもって8×8ドット，8×16ドットフォント表示モードを選択する必要があります。モード選択は画面管理I/Oポート(1FD0H)のL/H Resビット(D0)と$\overline{25}$/12行ビット(D2)によって図III-21のように行います。

図III-20 X1 turbo CGROM内文字フォント構造

図III-21　文字表示モード選択

$\overline{\text{L}}$/H Res.	$\overline{25}$/12行	フォント
0	0	8×8ドット
0	1	8×16ドット
1	0	〃
1	1	〃

CGROMフォントデータ読み出し

X1でCGROMからフォントデータを読み出すにはCRTCの動作を詳しく知る必要があります。ここでは読み出し方のプログラム例だけをあげることにして、後にPCGRAMのアクセスのところでCRTCの動作を説明します。この部分をよく読むとCGROMのアクセスおよびリストIII-7のプログラムを理解できるようになります。

リストIII-7

```
FONTDT  EQU     0B000H
        LD      BC,23E8H
        LD      A,18H
        LD      D,20H
ATTSET: OUT     (C),A
        INC     BC
        DEC     A
        JR      NZ,ATTSET
        LD      BC,33E8H
        LD      A,18H
        LD      D,41H
ASCSET: OUT     (C),D
        INC     BC
        DEC     A
        JR      NZ,ASCSET
        LD      BC,1400H
        LD      E,08H
        LD      HL,FONTDT
        EXX
        LD      BC,1A01H
VDSPL:  IN      A,(C)
        JP      P,VDSPL
VDSPL1: IN      A,(C)
        JP      M,VDSPL1
        DI
        EXX
READFT: IN      A,(C)
        LD      (HL),A
        INC     HL
        INC     BC
        NOP
        LD      A,0EH
DELAY:  DEC     A
        JP      NZ,DELAY
        DEC     E
        JP      NZ,READFT
        RET
```

CGROMからキャラクタ“A”(ASCIIコード41H)のフォントデータを読み出すプログラム例(リストIII-7)を示します。メインアドレスB000H番地以後にフォントデータを格納することにします。

他の文字のCGROMフォントデータを得たい場合はその文字のASCIIコードを41Hのところに設定してこのプログラムを実行すればよいのです。

X1 turboでも従来のX1と同じプログラムでCGROMからフォントデータを読み出すことができます。ただし，それには従来のプログラムの前にI/Oポート1FD*HのD5ビットを“0”に設定してから行います。ところがX1 turboでは従来の面倒で遅い方法をハードウェアの改良によってCGROMをソフトウェアから簡単にしかも高速にアクセスできるモードを持っています。以下はこの方法について説明します。

このモードでは，CGROMのフォントデータを読み出す際に図III-22の各設定を行う必要があります。

図III-22　CGROMフォントデータ読み出し設定値

I/Oポート	設　定　値
1FD*HのD5 ($\overline{\text{SPCG}}$/FPCG)	1
1FD*HのD6 (CGSEL $\overline{8}$/16)	0：8×8ドットフォント 1：8×16ドットフォント
27FFHのD5 ($\overline{\text{ROM}}$/RAM)	0
3FFFHのD7 ($\overline{\text{CG}}$/KANJI)	0
37FFH	読み出したい文字のASCIIコード

漢字テキスト，アトリビュートV-RAMの他のビット状態はCGROMのアクセスには無関係です。以下に先にX1の場合にあげたプログラムと同じ働きをするプログラム(リストIII-8)を示します。ただしこのプログラムは，“A”文字の8×16ドットフォントの読み出しになっています。

リストIII-8

```
FONTDT  EQU     0B000H

        LD      BC,1FD0H
        LD      A,*11*****B     ;D6, D5以外のビットは，画面モードに合わせて設定
        OUT     (C),A
        LD      BC,3FFFH        ┐
        LD      A,00000000B     │漢字テキストV-RAM
        OUT     (C),A           ┘
        LD      B,27H           ┐
        LD      A,00000000B     │アトリビュートV-RAM
        OUT     (C),A           ┘
        LD      B,37H           ┐
        LD      A,41H           │テキストV-RAMに“A”のASCIIコード
        OUT     (C),A           ┘
        LD      BC,1400H        ;CGROM I/Oポート
        LD      E,10H           ;16バイト読み出し
        LD      HL,FONTDT       ;フォントデータ格納先頭アドレス
READFT: IN      A,(C)
        LD      (HL),A
        INC     HL
        INC     C
        DEC     E
        JR      NZ,READFT
        RET
```

■PCGRAMのアクセス

PCGRAMの構成

X1シリーズではユーザーが定義可能なPCGRAMをもち，青，赤，緑の各色について2Kバイト計6Kバイトをもっています。8×8ドットフォントデータで256個の文字定義が可能で，またX1 turboでは8×16ドットフォントデータで128個の文字定義モードもあります。PCGRAMをアクセスするには次のシステムI/Oポートを使用します。

・青PCGRAM：15**H

・赤PCGRAM：16**H

・緑PCGRAM：17**H

PCGRAM文字表示

PCGRAMの文字表示方式はPCGキャラクタ方式とPCG外字方式(X1 turboのみ)のふたとおりがあります。

PCGキャラクタ方式

これは従来のX1のPCGRAM文字表示方式であり，表示されるフォントは8×8ドット構成になります。この方式では256種類のキャラクタ表示ができます。

この方式で，文字を表示させるためには画面上の位置に対応する各V-RAMのアドレスのデータを次のように設定します。

① テキストV-RAM：ASCIIコードを格納。

② アトリビュートV-RAM：$\overline{\text{ROM}}$/RAM選択ビット(D5)を1に設定。

③ 漢字テキストV-RAM(X1 turboのみ)：$\overline{\text{CG}}$/KANJIビット(D7)を0に，$\overline{1}$/2水準ビット(D4)を0に設定。

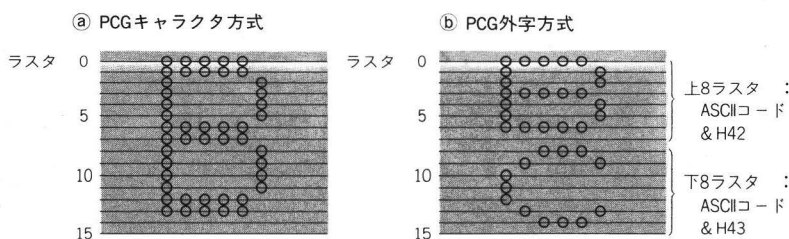
PCG外字方式—turboのみ

これはPCGRAMで定義した漢字(外字)などを表示するためにX1 turboだけにつけられた方式です。1行中に縦16ドット(8×16ドット)のフォントを表示させることができます。

この方式では，テキストV-RAMに書き込むASCIIコードは偶数，奇数の区別がなく偶数，奇数(偶数<奇数)の連続したASCIIコードを持つキャラクタは常に8×16ドットのフォントをもつひとつのキャラクタとみなされます。したがってテキストV-RAMにはどちらか一方のASCIIコードを書き込んでおけばすみます。

この方式ではふたつのキャラクタを1キャラクタとみなすので表示できるパターンは128種類になります。また，この方式は1行に16ラスタ以上のラインが必要ですので，低解像度モードの25(20)行モードではこのPCG外字方式の表示は使えません。ただし漢字ROM表示と同様，縦倍表示にすれば画面に出すことが可能ですがテキスト画面上の文字の大きさより縦に2倍大きくなります。

例として画面モードを低解像度モードの12(10)行モードか，高解像度モードの25(20)行

図Ⅲ-23　PCGRAM文字表示方式

モードにしておき，PCGRAMのASCIIコード42H，43HにCGROMと同じ『B』と『C』のデータを定義しておきます。図Ⅲ-23はテキストV-RAMにASCIIコードの42Hを格納したときのPCGキャラクタ方式と，PCG外字方式の表示結果を示します。ⓑのPCG外字方式では42H,43Hの2バイトで，BとCが8×16ドット構成の文字ひとつとみなされていることがわかります。

PCG外字方式で文字を表示するためには各V-RAMを次のように設定します。

① テキストV-RAM：ASCIIコードを格納(偶数，奇数のどちらでもよい)。

② アトリビュートV-RAM：$\overline{\text{ROM}}$/RAM選択ビット(D5)を1に設定。

③ 漢字テキストV-RAM ：$\overline{\text{CG}}$/KANJIビット(D7)か$\overline{1}$/2水準ビット(D4)の少なくともどちらかを1に設定。

PCGRAMフォントデータ定義，読み出し

CRTCによって指定できるテキストV-RAMは2048個ですが，実際に画面表示に使用するのは図Ⅲ-14のとおりです。1000文字画面モードの場合それぞれのページに24個，2000文字画面モードの場合は48個のテキストV-RAMが未使用になっています。

PCGRAMをアクセスする場合はこの未使用アドレスを使って行います。ただし，これを行うにはCRTCが上に述べた表示されないバイトをアクセスしている間，垂直帰線期間である必要があり，これは8255②のポートB_7(V-DISP信号)の立ち下がりを検出することで垂直帰線期間の開始を知ることができます。具体的にはリストⅢ-9のようにして行います。

リストⅢ-9

```
         LD      BC,1A01H
VDSPL:   IN      A,(C)
         JP      P,VDSPL
VDSPL1:  IN      A,(C)
         JP      M,VDSPL1
```

次に40文字画面モード時における青のPCGRAM(I/Oポート1500H)に対して『A』というパターンを定義するプログラム例(リストⅢ-10)を示します(ユーザ定義番号を61Hとした場合)。

PCGRAMへの書き込み，読み出しの前に必ず属性V-RAMの未使用エリアにCGROM/

PCGRAM モードのビットセットを行わなければなりません(D5 を 1 にする)。

読み出しの場合は[ ]の部分を以下のようにすればできます。

```
RDPCG:  IN      A,(C)
        LD      (HL),A
```

赤，緑 PCGRAM へのデータの書き込み読み出しも同様に行えます。また 1 回の垂直帰線期間に全色(青，赤，緑)PCGRAM へのデータの書き込み，読み出しを行うこともできますが，ここでは省略します。

リストIII-10

```
         LD      BC,23E8H          :23E8Hは垂直帰線期間の属性V-RAMスタートアドレス
         LD      A,18H             :未使用属性V-RAM 24バイトに対して設定
         LD      D,20H             :属性V-RAMのD5を“1”にする
ATTSET:  OUT     (C),D             ┐
         INC     BC                │未使用テキストエリア(24バイト)に対応した属性V-RAM
         DEC     A                 │へ20Hの書き込みにより，PCGRAMモードのビットを設定
         JR      NZ,ATTSET         ┘
         LD      BC,33E8H          :垂直帰線期間のテキストV-RAMスタートアドレス
         LD      A,18H             :未使用テキストV-RAM 24バイトに対した設定
         LD      D,61H
ASCSET:  OUT     (C),D             ┐
         INC     BC                │未使用テキストエリア(24バイト)に定義したい定義番号
         DEC     A                 │61Hを書き込む
         JR      NZ,ASCSET         ┘
         LD      BC,1500H          :青PCGRAMアクセスI/Oポート
         LD      E,08H             :ラスタカウンタ8バイト書き込み
         LD      HL,DATA
         EXX
         LD      BC,1A01H          ┐
VDSPL:   IN      A,(C)             │
         JP      P,VDSPL           │VDSPL信号立ち下りの検出
VDSPL1:  IN      A,(C)             │
         JP      M,VDSPL1          ┘
         DI
         EXX
SETPCG:  LD      A,(HL)            ┐
         OUT     (C),A             │
         INC     HL                │
         INC     BC                │
         NOP               ┐       │
         LD      A,0EH     │このディレイ │1ラスタごとにデータを書き込む
DELAY:   DEC     A         │タイムを正確 │
         LP      NZ,DELAY  ┘に守ること  │
         DEC     E                 │
         JP      NZ,SETPCG         ┘
;
;        (EXIT)
;
DATA:    DB      18H,24H,42H,7EH   書き込みデータ
         DB      42H,42H,42H,00H
```

X1 turbo では PCGRAM 文字表示と同様に PCGRAM を定義したり，読み出したりする方式にも PCG キャラクタ方式と PCG 外字方式があります。

PCGキャラクタ方式

これは 8×8 ドットのパターン(8 バイト)をアクセスする際に用いる方式で，従来の X1 と同じ動作です。

ただし X1 turbo ではハードウェアの改良によって従来の X1 の場合における CGROM の読み出しとか，PCGRAM アクセス時の面倒なソフトウェアとハードウェアの同期をプログラムで考慮する必要はなくなり，PCGRAM のアクセス時間が大幅に高速になっています。もちろん従来の X1 とコンパチのモードで CGROM, PCGRAM をアクセスすることもできます。I/O ポート 1FD * H の D5 ビットを 0 にすると従来のやり方とコンパチにな

ります。ここではX1 turboに特有の高速モードについて説明します。

この方式でPCGRAMをアクセスするには図III-24の各設定を行う必要があります。

漢字テキスト，アトリビュートV-RAMの他のビット状態はPCGRAMのアクセスには無関係です。

例として，青PCGRAMの41H番のユーザー定義文字に『A』のパターンを書き込むプログラム(リストIII-11)を示します。

図III-24　PCGキャラクタ方式アクセス設定値

I/Oポート	設　定　値
1FD＊HのD5 ($\overline{SPCG}$/FPCG)	1
27FFHのD5 ($\overline{ROM}$/RAM)	1
3FFFHのD7 ($\overline{CG}$/KANJI)	0
3FFFHのD4 ($\overline{1}$/2水準)	0
37FFH	アクセスしたいユーザー定義番号

リストIII-11

```
         LD      BC,1FD0H
         LD      A,**1*****B
         OUT     (C),A
         LD      BC,3FFFH
         LD      A,00000000B
         OUT     (C),A
         LD      BC,27FFH
         LD      A,00100000B
         OUT     (C),A
         LD      B,37H
         LD      A,41H
         OUT     (C),A
         LD      BC,1500H
         LD      E,08H
         LD      HL,DATBUF
PCGWRT:  LD      A,(HL)      ] 読み出す場合   IN A,(C)
         OUT     (C),A       ]                LD(HL),A
         INC     HL
         INC     C           ] Cのインクリメントは必ず2回
         INC     C           ] (C=C+2)でなければなりません
         DEC     E
         JR      NZ,PCGWRT
         RET

DATBUF:  DB      18H,24H,42H,7EH
         DB      42H,42H,42H,00H
```

読み出す場合はリストの右にあるようにプログラムを変更すれば良く，読み出しデータはDATBUF以後のアドレスに格納されます(8バイト)。

PCG外字方式

これは8×16ドットのパターン(16バイト)をアクセスする際に用います。

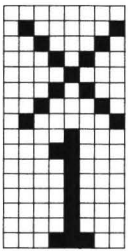
この方式でPCGRAMをアクセスするには図III-25の各設定を行う必要があります。

図III-25　PCG外字方式アクセス設定値

I/Oポート	設　定　値
1FD＊のD5（$\overline{SPCG}$/FDCG）	1
27FFHのD5（$\overline{ROM}$/RAM）	1
3FFFHのD7（$\overline{CG}$/KANJI）	少なくともどちらか一方を1
3FFFHのD4（$\overline{1}$/2水準）	
37FFH	アクセスしたいユーザー定義番号(0~255)

漢字テキスト，アトリビュートV-RAMの他のビット状態はPCGRAMのアクセスには無関係です。

37FFHに設定したユーザー定義番号が奇数の場合，その値−1の偶数から16バイト分のフォントデータがアクセスされます。偶数の場合はそのユーザー番号からのフォントデータがアクセスされます。例としてPCG外字方式でユーザー番号60H，61Hのところに，"X/1"のデータを定義するプログラムを図III-26に示します。

図III-26　"X/1"データ定義（PCG外字方式）

```
         LD      BC,1FD0H
         LD      A,**1*****B
         OUT     (C),A
         LD      BC,3FFFH
         LD      A,10010000B
         OUT     (C),A
         LD      B,27H
         LD      A,00100000B
         OUT     (C),A
         LD      B,37H
         LD      A,60H
         OUT     (C),A
         LD      BC,1500H
         LD      E,10H
         LD      HL,DATBUF
PCGWRT:  LD      A,(HL)
         OUT     (C),A
         INC     HL
         INC     C
         DEC     E
         JR      NZ,PCGWRT
         RET
```

```
DATBUF:  DB      00H
         DB      41H
         DB      22H
         DB      14H
         DB      08H
         DB      14H
         DB      22H
         DB      41H
         DB      08H
         DB      18H
         DB      08H
         DB      08H
         DB      08H
         DB      08H
         DB      08H
         DB      1CH
```

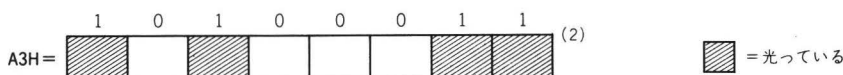
上のプログラムで定義した"X/1"のフォントを画面上すみに表示する例をリストIII-12に示します。

```
リストIII-12
LD       BC,3800H
LD       A,10010000B
OUT      (C),A
LD       B,20H
LD       A,00100111B
OUT      (C),A
LD       B,30H
LD       A,60H
OUT      (C),A
RET
```

■グラフィック画面

図III-28 に X1 および X1 turbo の画面上の位置とグラフィック V-RAM 番地の対応を示します。ひとつのマスは画面上の横 8 ドットのドット列を示しています。このドット列は 8 ビットの 2 進数の各ビットに対応して 1 バイトのデータを作ることになります。

たとえば I/O4000H (青 GRAM) に A3H を格納するということは画面左上隅に，図III-27 のようなパターンが青色で表示されることと同等です。ただし，パレット，プライオリティが青 GRAM 表示状態になっているとします。

図III-27　X1のグラフィックV-RAMデータと画面表示の関係

図III-27 のようなパターンが，縦に 8 個あるいは 16 個ならんでひとつの文字の大きさを形づくります。

グラフィック V-RAM は，青，赤，緑に分かれており，この 3 色の重ね合わせでドットごとにとる色が決まります。図III-28 中の I/O 番地はすべて青色の V-RAM を表しています。この番地に 4000H を加えたものが赤色面の V-RAM の番地を表し，さらに 4000H を加えたものが緑色の V-RAM の番地を表します。

図III-28　グラフィック画面とV-RAM番地の対応

X1の場合

1文字表示分（横）／1文字表示分（縦）

4000	4001		404E	404F
4800	4801		484E	484F
5000	5001		504E	504F
5800	5801		584E	584F
6000	6001		604E	604F
6800	6801		684E	684F
7000	7001		704E	704F
7800	7801		784E	784F
4050	4051		409E	409F
4850	4851		489E	489F
5050	5051		509E	509F
5850	5851		589E	589F
6050	6051		609E	609F
6850	6851		689E	689F
7050	7051		709E	709F
7850	7851		789E	789F
4780	4781		47CE	47CF
4F80	4F81		4FCE	4FCF
5780	5781		57CE	57CF
5F80	5F81		5FCE	5FCF
6780	6781		67CE	67CF
6F80	6F81		6FCE	6FCF
7780	7781		77CE	77CF
7F80	7F81		7FCE	7FCF

200ドット（8ドット×25）

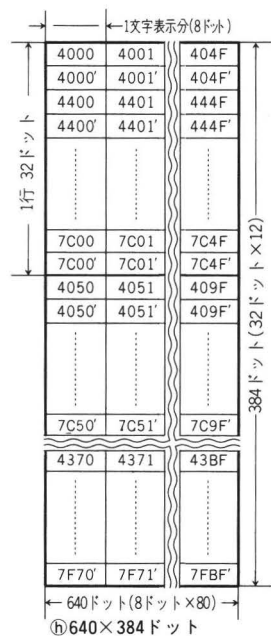
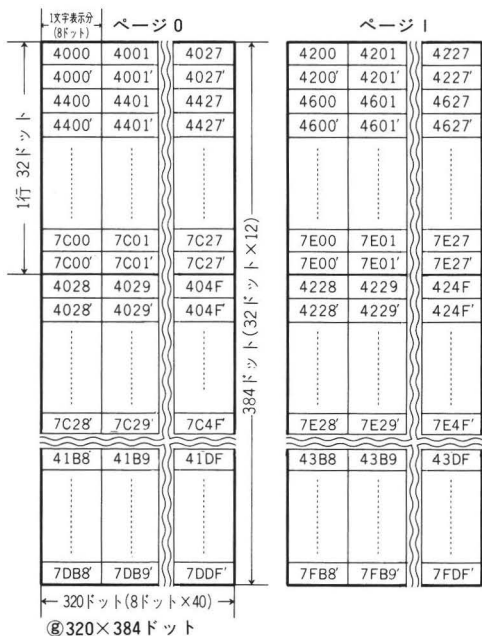
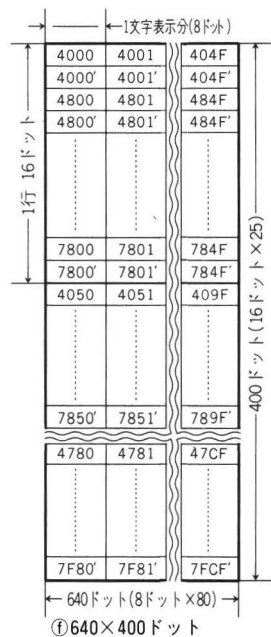
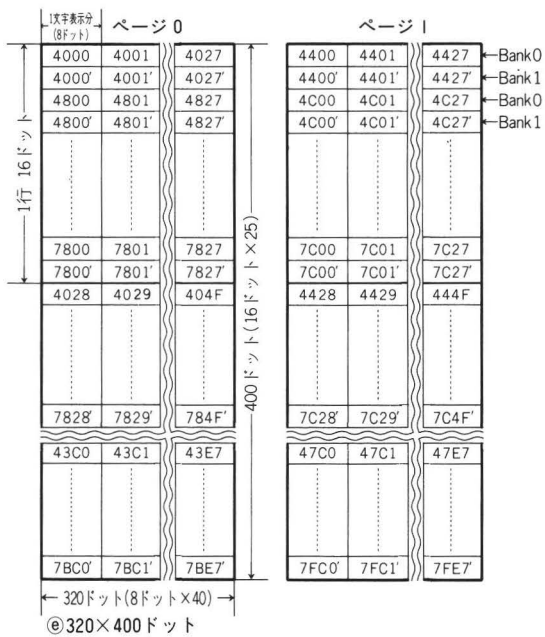
640ドット（8ドット×80）

ⓐ640×200ドット　　BLUE

ⓑ320×200ドット BLUE（ページ0）

ⓒ BLUE（ページ1）

X1 turboの場合

1文字表示分(8ドット)
ページ0
ページ1
4000 4001 4027
4800 4801 4827
5000 5001 5027
5800 5801 5827
7000 7001 7027
7800 7801 7827
4028 4029 404F
7828 7829 784F
43C0 43C1 43E7
7BC0 7BC1 7BE7
4400 4401 4427
4C00 4C01 4C27
5400 5401 5427
5C00 5C01 5C27
7400 7401 7427
7C00 7C01 7C27
4428 4429 444F
7C28 7C29 7C4F
47C0 47C1 47E7
7FC0 7FC1 7FE7
1行 8ドット
200ドット(8ドット×25)
320ドット(8ドット×40)
ⓐ320×200ドット
1文字表示分(8ドット)
4000 4001 404F
4800 4801 484F
5000 5001 504F
5800 5801 584F
7000 7001 704F
7800 7801 784F
4050 4051 409F
7850 7851 789F
4780 4781 47CF
7F80 7F81 7FCF
640ドット(8ドット×80)
ⓑ640×200ドット
4000 4001 4027
4400 4401 4427
4800 4801 4827
4C00 4C01 4C27
7C00 7C01 7C27
4028 4029 404F
7C28 7C29 7C4F
41B8 41B9 41DF
7DB8 7DB9 7DDF
4200 4201 4227
4600 4601 4627
4A00 4A01 4A27
4E00 4E01 4E27
7E00 7E01 7E27
4228 4229 424F
7E28 7E29 7E4F
43B8 43B9 43DF
7FB8 7FB9 7FDF
1行 16ドット
192ドット(16ドット×12)
320ドット(8ドット×40)
ⓒ320×192ドット
4000 4001 404F
4400 4401 444F
4800 4801 484F
4C00 4C01 4C4F
7C00 7C01 7C4F
4050 4051 409F
7C50 7C51 7C9F
4370 4371 43BF
7F70 7F71 7FBF
640ドット(8ドット×80)
ⓓ640×192ドット

注1. 400ドットおよび384ドットモード時は，グラフィックV-RAM BANK0，1の内容を1ラスタごと交互に画面に表示する。
2. ダッシュ付はGRAM BANK1

画面の操作

■画面の切り換え

X1の切り換え操作

画面モード切り換え

すでにCRTコントローラのところで説明した40文字モードと80文字モードの設定によりテキスト画面モードを切り換えることができます。

グラフィック画面は，

40文字モード↔320×200ドット(2ページ)

80文字モード↔640×200ドット

と対応しているので，テキスト画面に対する設定がそのままグラフィック画面に対する設定を意味しています。

表示画面ページの切り換え

画面モードが40×25文字320×200ドットのときのみ，画面を2ページ持っているのでどちらをディスプレイに表示するか選択することになります。

表示ページの変更はCRTCのレジスタR12を変えるだけですみます。たとえば0ページから1ページにするにはリストIII-13のようになります。

リストIII-13

```
LD      BC,1800H ┐
LD      A,0CH    │ CRTCのR12レジスタを選択
OUT     (C),A    ┘
INC     BC       ┐
LD      A,04H    │ レジスタにスタートアドレスをセット
OUT     (C),A    ┘
```

要するに次のようにすればよいのです。

0ページ→1ページ　　R12に04Hをセット

1ページ→0ページ　　R12に00Hをセット

なお画面モードや表示画面ページにかかわらずその表示画面で色単位に表示を切り換えるには後に述べるパレット機能やプライオリティ機能を使うので参照してください。

X1 turboの切り換え操作

画面モードの切り換え

画面モードの切り換えは，次のI/Oに対する3つの処理により実行されます。

① CRTCレジスタの設定

1800H：レジスタ番号のセット

1801H：データのセット

② 8255②の設定

ポートCのD6(I/Oポート1A03H)：40/$\overline{80}$文字モード

③ 画面管理用I/Oポート

1FD*H：ビット0，1，2，7

①のCRTCレジスタの設定法についてはすでに述べました。設定値についても図III-10に示しています。クロック周波数の切り換えは，次に説明する8255②の設定によりセットされるので考える必要はありません。

②の8255②の設定はI/Oポート1A03Hに対する設定値でクロックが変えられるようになっています。40文字モードから80文字モードへの変更は次のようにします。

```
LD      BC,1A03H
LD      A,0CH
OUT     (C),A
```

Aレジスタにセットした0CH＝00001100(2)で，第0ビットを1にすると40文字モードへの切り換えになります。つまり，

1A03Hへの設定値 { 0CH……80文字モード / 0DH……40文字モード }

ということです。

③の画面管理用I/Oポート(1FD*H)の内容を図III-29で示します。

ここで関係するのはビット0，1，2，7の計4ビットです。注意しなければいけない

図III-29　画面管理用I/Oポート：1FD*H

ビット名	コントロール内容	
D0	$\overline{L}$/H Res.	0：低解像度モード（200ドット表示） 1：高解像度モード（400ドット表示）
D1	$\overline{1}$/2RA	0：グラフィック表示＝320(640)×400(384)ドットモード 1：グラフィック表示＝320(640)×200(192)ドットモード
D2	$\overline{25(20)}$/12(10)行	0：テキスト表示＝25(20)行モード 1：テキスト表示＝12(10)行モード
D3	DISP Bank $\overline{0}$/1	0：G-RAM Bank 0を表示 1：G-RAM Bank 1を表示
D4	CPU Bank $\overline{0}$/1	0：G-RAM Bank 0をCPUアクセス 1：G-RAM Bank 1をCPUアクセス
D5	$\overline{SPCG}$/FPCG	0：コンパチCGアクセスモード(X1とのコンパチ) 1：高速CGアクセスモード
D6	CGSEL $\overline{8}$/16RA	0：8 RA/CHRのCGフォントをCPUアクセス 1：16RA/CHRのCGフォントをCPUアクセス
D7	$\overline{25(12)}$/20(10)行	0：テキスト表示＝25(12)行 1：テキスト表示＝20(10)行or10行(アンダーライン表示)

のはこのI/Oポートが書き込み専用であるということです。ですから他のビットに影響が出ないように設定してください。BIOS ROMではメモリF8D6H番地をこのコントロールワードのバッファとして使っています。このためユーザーもこれを利用し，同時にこの値も書き直すとよいでしょう。

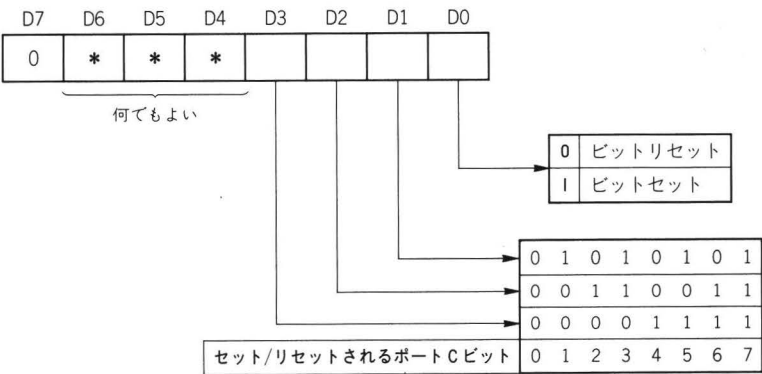
各画面モード時の画面管理用I/Oポートの設定値を図III-30, 31に示します。

図III-30　各表示画面モードの管理I/Oポート(IFD＊H)の設定値(低解像度モード)

テキスト画面	40×25	80×25	40×12	80×12	40×20	80×20	40×10	80×10
グラフィック画面 / データ	320×200	640×200	320×192	640×192				
D0	0	0	0	0	0	0	0	0
D1	0	0	0	0	＊	＊	＊	＊
D2	0	0	1	1	0	0	1	1
D3	＊	＊	＊	＊	＊	＊	＊	＊
D4	＊	＊	＊	＊	＊	＊	＊	＊
D5	＊	＊	＊	＊	＊	＊	＊	＊
D6	＊	＊	＊	＊	＊	＊	＊	＊
D7	0	0	0	0	1	1	1	1

column 5

8255のビット制御

並列インタフェースIC8255において，ポートCが出力に指定されているときにはそのうちの任意の1ビットだけをセットまたはリセットすることができます。

したがって本文の例で，D6をセット（あるいはリセット）するのは，

00001100(2)　(＝0CH)　……80文字
D6 リセット

00001101(2) (＝0DH)　……40文字
D6 セット

となります。

図III-31　各表示画面モードの管理I/Oポート(1FD＊H)の設定値(高解像度モード)

テキスト画面	40×25	80×25	40×12	80×12	40×25	80×25	40×12	80×12	40×20	80×20
グラフィック画面 / データ	320×200	640×200	320×192	640×192	320×400	640×400	320×384	640×384		
D0	1	1	1	1	1	1	1	1	1	1
D1	1	1	1	1	0	0	0	0	＊	＊
D2	0	0	1	1	0	0	1	1	0	0
D3	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊
D4	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊
D5	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊
D6	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊
D7	0	0	0	0	0	0	0	0	1	1

表示画面ページの切り換え

テキスト画面とグラフィック画面の対応

各画面モードにおけるV-RAMの使用状況とテキスト画面とグラフィック画面がどのように対応して表示されるかを図III-32に示します。

ページの切り換え

図III-32に示した中で右側の欄の画面(ⓐ～)のうち表示させる画面を切り換えるのがページ切り換えです。各モードにおいてページの切り換えに必要な操作を図III-33に示します。

図III-33中の切り換え操作A～Cは，

A：下記のⓐを実行

B：下記のⓑを実行

C：下記のⓐとⓑを実行

を意味します。

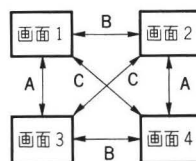
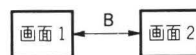
ⓐ：CRTコントローラのレジスタR12にメモリアドレス先頭番地の上位バイトを設定。
(I/Oポート1800H，1801H)

ⓑ：画面管理用I/Oポートにグラフィック RAMの表示バンク切り換え(DISP Bank $\overline{0}$/1)を設定。
(I/Oポート1FD＊HのD3)

各画面モードにおける各ページに対する設定データを図III-34に示します。

図III-34の中の1FD＊H，D3の項の0と1はD3ビットについてだけリセットするかセットするという意味であり，そのポートに00Hあるいは01Hを送ることを意味してはいないことに注意してください。

図III-33　ページ切り換えモード

① 40×25(12)，320×200(192)モード

② 80×25(12)，640×200(192)モード

③ 40×25(12)，320×400(384)モード

④ 80×25(12)，320×400(384)モード
ページはひとつ，切り換えはない

図Ⅲ-32　各画面モードにおけるV-RAMの使用状況

画面モード	V-RAMの分割状態	画面表示ページ
40×25 (12) 文字 320×200 (192) ドット	テキスト: ページ0 / ページ1 グラフィック: Bank 0 (ページ0 / ページ1), Bank 1 (ページ2 / ページ3)	ⓐ 画面1 テキスト ページ0, グラフィック ページ0 ⓑ 画面2 テキスト ページ0, グラフィック ページ2 ⓒ 画面3 テキスト ページ1, グラフィック ページ1 ⓓ 画面4 テキスト ページ1, グラフィック ページ3
80×25 (12) 文字 640×200 (192) ドット	テキスト: ページ0 グラフィック: Bank 0 (ページ0), Bank 1 (ページ1)	ⓐ 画面1 テキスト ページ0, グラフィック ページ0 ⓑ 画面2 テキスト ページ0, グラフィック ページ1
40×25 (12) 文字 320×400 (384) ドット	テキスト: ページ0 / ページ1 グラフィック: Bank 0 (ページ0 / ページ1), Bank 1 (ページ0 / ページ1)	ⓐ 画面1 テキスト ページ0, グラフィック ページ0 ⓑ 画面2 テキスト ページ1, グラフィック ページ1
80×25 (12) 文字 640×400 (384) ドット	テキスト: ページ0 グラフィック: Bank 0 (ページ0), Bank 1 (ページ0)	ⓐ 画面1 テキスト ページ0, グラフィック ページ0

ⓐの操作とⓑの操作プログラムをリストIII-14に示します。DATA1はCRTCのR12に設定する値で，DATA2はI/Oポート1FD＊HのD3(DISP Bank 0/1)に設定する値です。それぞれの画面モード，ページ値に合わせて設定します。

リストIII-14

```
LD      BC,1800H  ┐
LD      A,0CH     │
OUT     (C),A     │ ⓐ
INC     BC        │
LD      A,DATA1   │
OUT     (C),A     ┘


LD      BC,1FD0H  ┐
LD      A,DATA2   │ ⓑ
OUT     (C),A     ┘
```

図III-34　画面モード，ページ設定値

画面＼モード		40×25	40×12	80×25	80×12	40×25	40×12	80×25	80×12	40×20	40×10	80×20	80×10
		320×200	320×192	640×200	640×192	320×400	320×384	640×400	640×384				
画面1	CRTC, R12	00	00	00	00	00	00	00	00	00	00	00	00
	FD＊H, D3	0	0	0	0								
画面2	CRTC, R12	00	00	00	00	04	02			04	02		
	1FD＊H, D3	1	1	1	1								
画面3	CRTC, R12	04	02										
	1FD＊H, D3	0	0										
画面4	CRTC, R12	04	02										
	1FD＊H, D3	1	1										

グラフィックV-RAMのバンク切り換え

X1 turboではグラフィックV-RAMを2バンク持っており，表示するバンクを切り換えるには表示画面ページの切り換えで述べたⓑの操作つまりI/O 1FD＊HのD3ビットをセット/リセットすることが必要でした。

ところがこの切り換えはCRTコントローラに関するものであり，Z80CPUからのアクセスとは独立なものです。つまり画面上に表示しているグラフィックV-RAMとは別のバンクでCPUが読み出し/書き込みを行うことができるのです。

それではそのふたつの操作法を図III-35でまとめておきます。

図III-35　バンク切り換え設定値

CPUアクセス	画面表示	IFD＊, D4	IFD＊, D3
Bank 0	Bank 0	0	0
Bank 0	Bank 1	0	1
Bank 1	Bank 0	1	0
Bank 1	Bank 1	1	1

■特殊な画面コントロール

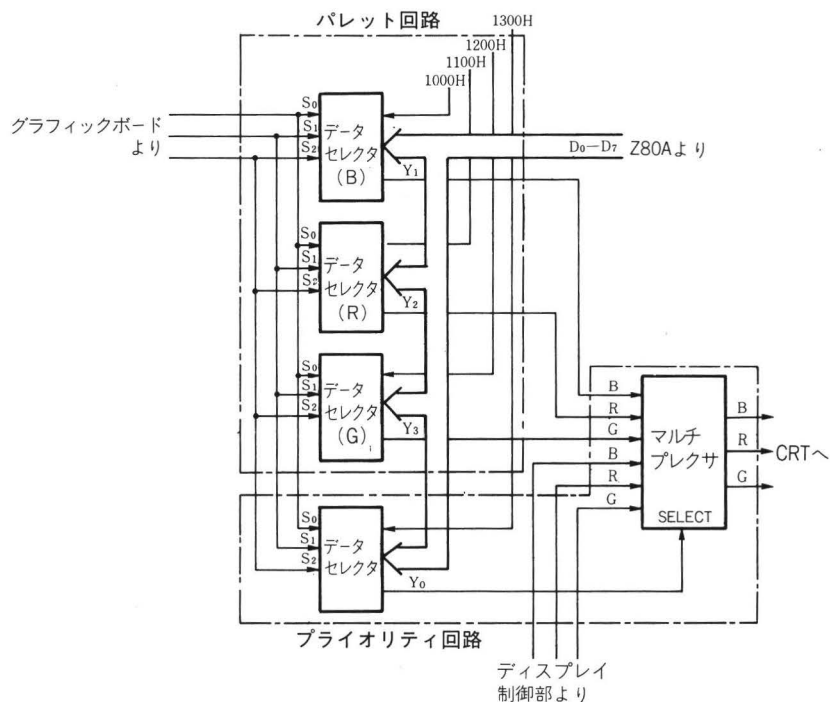
パレット機能

パレット機能とは，各色のグラフィックV-RAMから出力されるデータの内容をいちいち書き換えることなく，表示色を瞬間的に別の色に変えてしまう機能のことです。

パレット機能および次に説明するプライオリティ機能は，グラフィックV-RAMの内容をそのままCRTに出力せず，図III-36に示すパレット回路とプライオリティ回路を経ることにより実現されます。

パレット機能はどのように実現できるか図III-36を見ながら説明します。

RGBの3つのグラフィックV-RAMから出力された3ビットデータによって8とおりのカラーコードが表現できます。この3ビットのデータは回路図のようにパレット回路を構成する3つのデータセレクタとプライオリティ回路を構成するひとつのデータセレクタの入力となります。ここではパレット回路の3つのデータセレクタだけに注目します。これはそれぞれは青，赤，緑のデータセレクタとして働きます。これらのデータセレクタに3ビットのカラーコードが入力するとその値に応じて各データセレクタから図III-37の値がそれぞれY_3，Y_2，Y_1から出力されます。たとえばS_2，S_1，S_0がそれぞれ0，1，1だと各データセレクタのD3がそれぞれY_3，Y_2，Y_1に出力されます。この3つのY_3，Y_2，Y_1が実際の色を合成しパレットコードになります。たとえばY_1(青)＝1，Y_2(赤)＝0，Y_3(緑)＝0ですとそのとき入力したカラーコードの実際の色は青になります。またY_1＝1，

図III-36　パレット回路とプライオリティ回路ブロック図

図III-37　パレット回路のデータセレクタ

カラーコード	S_2(G)	S_1(R)	S_0(B)	パレットコード データセレクタ			パレットコード データセレクタ(初期値)			初期色
				G (12**H)	R (11**H)	B (10**H)	G	R	B	
0	0	0	0	D0	D0	D0	0	0	0	黒
1	0	0	1	D1	D1	D1	0	0	1	青
2	0	1	0	D2	D2	D2	0	1	0	赤
3	0	1	1	D3	D3	D3	0	1	1	マゼンタ
4	1	0	0	D4	D4	D4	1	0	0	緑
5	1	0	1	D5	D5	D5	1	0	1	シアン
6	1	1	0	D6	D6	D6	1	1	0	黄色
7	1	1	1	D7	D7	D7	1	1	1	白
				Y3	Y2	Y1	↓ F0H	↓ CCH	↓ AAH	

$Y_2=1$，$Y_3=1$ですと実際の色は青＋赤＋緑＝白になります。

図III-37の中の各データセレクタのD0～D7の値はソフトウェアによって設定でき，それぞれ図III-38のパレット回路のI/Oポートを介して値を設定します。

このデータを操作することにより，カラーコードに対するパレットコードを瞬間的に変えることができるわけです。通常カラーコード(S_2，S_1，S_0の組み合わせ)とパレットコード(Y_3，Y_2，Y_1)が一致するように各データセレクタにはF0H(緑)，CCH(赤)，AAH(青)を初期値に設定します。リストIII-15にこれを行うプログラム例を示します。

図III-38　パレット回路のデータセレクタ値

データセレクタ	I/Oポート
B	10**H
R	11**H
G	12**H

リストIII-15

```
LD      BC,1000H
LD      A,0AAH
LD      D,0CCH
LD      E,0F0H
OUT     (C),A
INC     B
OUT     (C),D
INC     B
OUT     (C),E
RET
```

ここでたとえばカラーコード3に黄色を対応させたいとします。黄色はRGB＝(110)つまり赤と緑の合成色ですから，図II-37中のパレットコード(データセレクタ)G,R,BのD3ビットをそれぞれ1,1,0とすればよいのです。

BASICでこれを実現するには，

```
PALET 3,6
```

とするだけですが，マシン語の場合はBASICのようにカラーコードごとに設定するのではなく，データセレクタごとに設定する必要があります。

カラーコードに対するパレットコードをひとつ変更するのに3つのI/Oポートすべての対応するビットを書き直す必要があることに注意してください。

プライオリティ機能

テキスト画面上の文字などとグラフィック画面上の図形が重なった場合に，遠近関係をどのように表示するかの指定ができる機能をプライオリティ機能といいます。単に両画面間単位で優先度が決められるのではなく，グラフィック画面上の各8色のデータごとにテキスト画面との優先を決めることができます。

column 6

パレット機能による画面モードの使いわけ

グラフィックRAMは，R,G,B1画面(320×200のときは2画面)ずつ，計3画面あります。この画面を合成することにより，ドットごとに8色を実現することができます。このほかパレット機能によって，各画面を8色中の任意の1色に割り振り，独立（重ね合わせない）して表示する方法もあります。

このように考えれば，グラフィック画面は，次の4つのどれか(X1 turboでは，200ドット/25行モードに対応します）を設定することができると考えられます(カッコ内はX1 turbo)。

ドットごとの8色指定モード

① 640×200ドット　1(2)面（WIDTH80:SCREEN,,0)
② 320×200ドット　2(4)面（WIDTH40:SCREEN0,0,0)
　　　　　　　　　　　　(WIDTH40:SCREEN1,1,0)

面単位の8色指定モード

③ 640×200ドット　3(6)面（WIDTH80:SCREEN,,n) {n=1,2,3}
④ 320×200ドット　6(12)面(WIDTH40:SCREEN0,0 n) {n=1,2,3}
　　　　　　　　　　　　(WIDTH40: SCREEN1,1,n) {n=1,2,3}

ドット単位に8色指定モードの場合は，表示データは表示する位置に対応する3つのグラフィックV-RAMのアドレスに同時に指定色に応じて格納します。

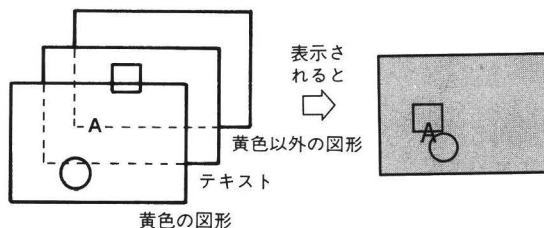
たとえば640×200ドット画面で，画面上左すみに黄色（赤と緑の合成）で1ドットを表示するように指定するには，グラフィックV-RAMの

4000H（Blue）のD7に0
8000H（Red）のD7に1
C000H（Green）のD7に1

のデータを格納します。

一方，面単位で8色指定モードの場合は，表示データは，R,G,Bのどれかの V-RAMにのみ格納します。このとき表示画面の色は自由に変えられますが，ある一瞬では1色だということです。

たとえば黄色だけをテキスト画面より優先するとします。そうすると図III-39のように黄色で書かれた図形(円)が文字(A)より上にあるように表示されるほか，黄色以外のカラーで書かれた図形(四角)はいちばん後ろにあるように見えます。

図III-39　黄色優先例

この機能は先に示した図III-36のプライオリティ回路により実現されます。マルチプレクサにより各ドットについてグラフィック画面のデータを出力するか，テキスト画面のデータを出力するかを振り分けます。それを決定するのがマルチプレクサのSELECT信号つまりグラフィック画面の各色に対応するデータセレクタの出力です。

この設定はI/Oポートの13＊＊Hに図III-40に示されるようなデータを書き込むことにより行います。

図III-40　プライオリティの設定データ

D7	D6	D5	D4	D3	D2	D1	D0
白	黄	シアン	緑	マゼンタ	赤	青	黒

：カラーコード

各ビットともテキスト画面より優先させるとき　1
〃　　　　　　　　優先させないとき 0

たとえば黄と緑だけをテキスト画面より優先させる場合はデータは，

$01010000_{(2)}=50H$

となり，プログラムは次のようになります。

```
LD      B,13H
LD      A,50H
OUT     (C),A
```

なお説明の便宜上，黄や緑などとしましたが，これはパレット機能が初期化されている場合であり，もし設定し直してあるのならば，そのカラーコードに対応させているパレットコードの色を意味することになりますので注意してください。

BASICではPRWコマンドがこのプライオリティ機能をサポートしています。

スーパーインポーズ機能

スーパーインポーズとはビデオやテレビ放送の画面とコンピュータ画面を重ねてディスプレイ上に表示させることをいいます。

BASICではディスプレイのモードの切り換えをCRTステートメントで行います。

図III-41　CRTn

nの値	ディスプレイの状態
0	テレビ映像を表示
1	コンピュータ画面を表示
2	テレビ映像(コントラストダウン)とコンピュータ画面を重ねて表示
3	テレビ映像(コントラストノーマル)とコンピュータ画面を重ねて表示

BASICを起動しなくてもキーボードから直接コントロールすることも可能です。たとえばCRT2は[SHIFT]＋[＋]により実行されます(図III-41)。

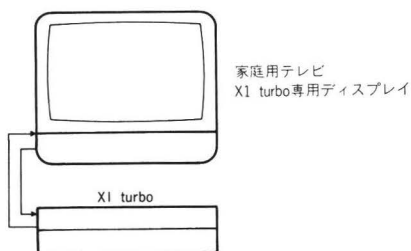
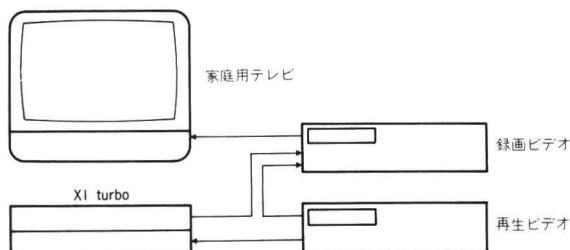
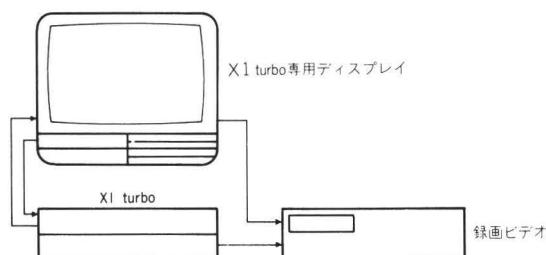
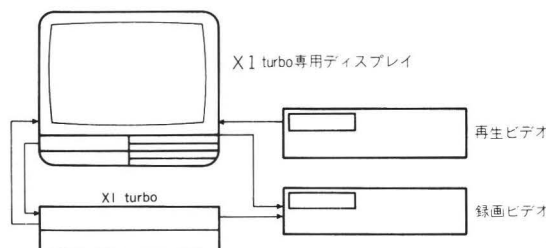
テレビ映像やビデオ，VHDなどの画像は通常NTSCコンポジットビデオ信号として出力されます。これは映像(色，輝度)信号と複合同期信号が重なったものです。

ディスプレイ上でこの映像とパソコン画面を重ねるために，NTSC規格の信号に同期を取りながらコンピュータ画面信号を送り出しています。これはX1の同期信号とNTSC信号の位相差を検出し，その差に対応する時間だけCRTコントローラを停止させることにより実現しています。ASC(自動同期制御)回路がこの操作を行っています。

X1 turbo以外のX1シリーズではスーパーインポーズは専用ディスプレイが必要です。またスーパーインポーズ画面をビデオ録画するには別売のデジタルテロッパが必要でした。X1 turboにはテロッパが内蔵されており，簡単にスーパーインポーズを応用してビデオの編集などができるようになりました。

X1 turboとディスプレイテレビの接続例を図III-42に4つ示します。図中で再生ビデオとあるところはビデオディスク，ビデオカメラなどとしてもかまいません。

図III－42　X1 turboにおけるスーパーインポーズの実現例

① テレビ放送画像とコンピュータ画像のスーパーインポーズ

② ビデオ画像あるいはテレビ放送画像をコンピュータ画像とスーパーインポーズして録画

③ コンピュータ画像またはテレビ放送画像とのスーパーインポーズを録画

④ ビデオ画像あるいはテレビ放送画像をコンピュータ画像とスーパーインポーズして録画(X1 turboの場合)

黒色制御機能

X1 turboのみに装備された機能です。他のX1シリーズにおいてはテキスト画面で黒色の表示は不可能でした。つまり黒色ではなく背景色がそのまま出てしまうわけです。このためパレットとプライオリティの機能により，テキスト画面を後ろに持っていって，みかけ上黒色にします。ところがスーパーインポーズ時にはテレビ映像が見えてしまい，不都合な場合も少々ありました。

X1 turboは画面の信号をカットする黒色状態を実現するための回路をプライオリティ回路に付加して黒色制御機能を備えています。ただし，これはX1 turbo専用ディスプレイが必要です。さらにこの機能を利用してテレビ画面のふちどりを黒く塗ることも可能にしました。

テキスト画面はカラーコード0以外の7色のうち任意の1色の文字を黒にすることができます。一方グラフィック画面ではカラーコード0とカラーコード1(青)の2色までを黒に変換することができます。ただし，グラフィック画面の場合は，パレットコードの設定が必要になります。

黒色制御はI/Oの1FE＊Hに図III-43に示すデータを送ればよいのです。たとえばテキストのシアンを黒変換するには次のようにします。

図III-43 I/Oポート1FE＊Hに送るコントロールデータの内容

1FE＊H	コントロール内容	
D0	テキスト(青)	黒変換するテキスト色の指定
D1	テキスト(赤)	
D2	テキスト(緑)	
D3	テキストの黒変換有効	
D4	グラフィック(カラーコード0)黒変換	
D5	グラフィック(カラーコード1)黒変換	
D6	テレビ画面のふちどりを黒変換	
D7	＊	

```
LD      BC,1FE0H
LD      A,0DH
OUT     (C),A
```

グラフィック画面の黒変換はさらにパレットコードの設定が必要となります。つまり黒変換したい色(ブルー，透明)のパレットコードを0にしなくてはなりません(図III-44)。

それでは透明を黒変換しさらにふちどりも黒くする方法(リストIII-16)を示しておきましょう。格納するデータの意味は次のとおりです。

図III-44

	D6	D5	D4	D3	D2	D1	D0
50H＝	1	0	1	0	0	0	0
	ふちどり		グラフィック(透明) ↓ 黒				

リストIII-16

```
LD      BC,1FE0H
LD      A,50H
OUT     (C),A
LD      A,0A9H
LD      D,0CCH
LD      E,0F0H
LD      BC,1000H
OUT     (C),A
INC     B
OUT     (C),D
INC     B
OUT     (C),E
```

アンダーライン表示機能

X1 turboには特定の条件のもとでアンダーラインを表示する機能があります。アンダーラインを表示することのできる画面モードは次のとおりです。

低解像度：40(80)×20行モード
　　　　　40(80)×10行モード
高解像度：40(80)×20行モード

このモードに設定する方法はすでに述べました。漢字テキストV-RAMの対応する位置のメモリデータのD5ビットをセットすればよいのです。

```
LD      BC,Kanji Address
IN      A,(C)
SET     5,A
OUT     (C),A
```

アンダーラインのカラーコードは1(青)を割り当てているので，その色を変えたい場合は，パレット機能により青を別の色に変えてください。

なおアンダーラインを表示できるモードではグラフィックを表示することはできないということと，アンダーラインはプリンタに表示されないことに注意してください。

サブCPUの働きとコントロール

サブCPUの機能構成

■サブCPUの意味

X1はメインCPU(Z80A)まわりの回路のほかに，I/O関係の処理のために別系統の回路を持っています。

サブCPU(80C49)を中心に，キーボード処理用IC(80C48，X1 turboでは80C49)，インタフェースIC(8255①)，タイマーIC(μPD1990)，ディスプレイテレビおよび電磁メカカセットで形成されるシステムです。

これらのIC類の電源は，メインCPU関連とは別の電源Vcc2に接続されています。この電源はフロントスイッチを切った後も通じており，機能の一部がなお有効です。

X1は，サブCPUの使い方が実に効率的です。なぜならば今までの多くのサブCPUやDMAを使った機種によくみられるメインCPUの効率の低下が見られないからです。それどころかサブCPUが，メモリ効率，処理速度の向上に多大な貢献をしています。

その意味でサブCPUの使い方として，同じ価格帯にある8ビット汎用マイコンの中で，ベストであるといえるでしょう。

X1・80C49サブCPUの受け持ちは次のとおりです。

1．キーボード
2．カセット
3．タイマー
4．カレンダ
5．テレビコントロール

これらの多くの分担作業を，メインCPUから離れて単独で処理するために，高インテリジェント化されています。

とくにキーボード読み込みには，80C48/49を利用してキーマトリクスのデコードが行われます。また割り込みをうまく用いて，メインCPUのソフト負担の大幅な軽減，処理速度の向上を図っています。

80C48(キーボードIC)と80C49はともにCMOS構造のICで，消費電力も少ないです。これらのICへの電源は，本体前面のスイッチを切っても流れていて，タイマーカレンダのカウントとモニタテレビのコントロールが常に行われます。

また，キーボード割り込みと，カセット状態の検出は割り込みによる方法と必要に応じ

て読み出す方法のふたつが自由に選べるため，プログラムの目的によって区別して使うことができます。

■サブCPU周辺の構成

サブCPUの概略と周辺IC

サブCPU周辺回路の構成概念図(図III-45)を示します。サブCPUを中心としたモジュールは，メインCPU側とは独立に動作しており一定の通信方法により会話を行います。

インタフェースIC8255は2個使われています。μPD1990はタイマーICです。

各ICについてその働きを見ていきましょう。

80C49(MSM80C49)

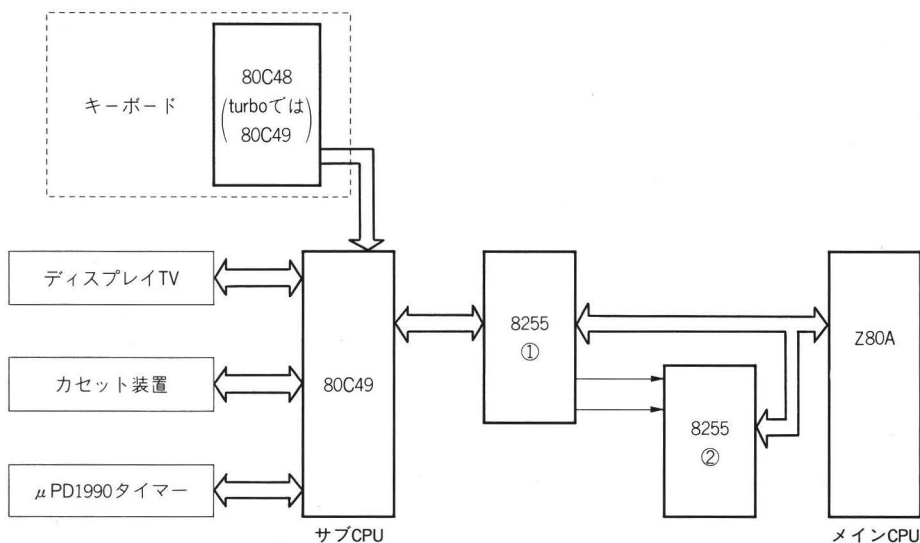
メインCPU(Z80A)をサポートするサブCPUです。キーコード処理，テレビ画面制御，テレビチャンネル，音量コントロール，カセットコントロール，時刻やカレンダの設定と読み出しを担当しています。

メインCPUから独立しているため，メインCPUがプログラム実行中でもカセット制御(APSSなども含む)とテレビ制御が自由に行えます。

また，IOCSルーチンの章で述べるソフトによるキーバッファとは別に，サブCPU内に8バイト(ファンクションデータを含めると16バイト)までのキー入力を蓄えることができます。

80C49は8080系のCPUで，内部に2Kバイトのマスク ROM，128バイトのRAM，I/Oポートなどを持つワンチップのCMOSマイクロコンピュータです。

このうちX1に用いられているのはMSM80C49というセカンドソースです。80C49はマスクROMを内蔵しているので，プログラムは製造時で決まってしまい外部からプログラ

図III-45　サブCPU周辺回路概念図

ミングすることはできません。

80C48 と 80C49 はほとんど同じ IC ですが，80C49 のほうが ROM, RAM とも 80C48 の 2 倍に強化されています。

80C48(キーボードIC)

X1では，キーボード内にあってキーマトリクスのデコードとキーデータのシリアル変換を担当しています。キーコードからASCII コードへの変換は,80C48 によって行われます。

したがって Z80A 側でキーコード，ASCII コード変換テーブルを用意する必要がなくなり，その分ソフトの負担が減ります。

80C48(キーボード)と 80C49(サブ CPU)との間は，わずか 3 本の線 (GND, Vdd, KEY DATA)で接続されて,KEYDATA にはシリアル変換されたキーデータが乗ります。X1 turbo ではこの IC のかわりに 80C49 が使われています。

キーデータの構成

キーデータ(キーボード用 IC とサブ CPU 間)は，ファンクションコード 1 バイトと，ASCII コードに変換されたキーコード 1 バイトの 2 バイトから成っています。

80C48 からはファンクションコード，キーコードの順で繰り返し送られてきます。ファンクションコードの各ビットの意味は図III-46 の表のとおりです。また，キーコード部の ASCII コード表を図III-47 に示します。

ここで注意しなければならないのは，キーデータ 2 バイト中の 1 バイト目のファンクションコードの第 7 ビットが 0 の場合です。bit7＝0 というのは，テンキー，ファンクションキー，テレビキー，カセットキーのいずれかが押されたことを意味します。この場合，キーコード部の持つ意味が変化します。

bit7＝0 の場合のキーコード部のコード表は図III-48 のとおりです。

80C49 からは，テンキーもテレビキーも区別なく割り込みがかかるので，ソフト側で第 7 ビットを見て不要なデータは破棄するなどの処理が必要です。また専用モニタを使用しない場合，チャンネルキーやボリュームキーもファンクションキーとして利用することができます。

図III-46　ファンクションコードのビット構成

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
	ファンクション	キーデータが有効/無効	リピート	GRAPH	CAPS	カナ	SHIFT	CONTL
"0"	・テンキー ・ファンクションキー ・TVキー ・カセットキー	・キーコード(8ビット)が有効である ヌルコード"00"以外が送られてきたとき	・リピートデータである	・GRAPHキーが押されている	・CAPSキーが押されている (LOCKされている)	・カナキーが押されている (LOCKされている)	・SHIFTキーが押されている	・CTRLキーが押されている
"1"	・上記以外	・キーコード(8ビット)が無効である ヌルコード"00"が送られてきたとき	・1回目のデータである	・GRAPHキーがはなされている	・CAPSキーがはなされている	・カナキーがはなされている	・SHIFTキーがはなされている	・CTRLキーがはなされている

図III-47　キーコード部のASCIIコード表

CTRLと同時に押し下げ（列0・1、列4・5）　GRAPHを押し下げしながら対応するキーを押し下げ（列8・9、列E・F）

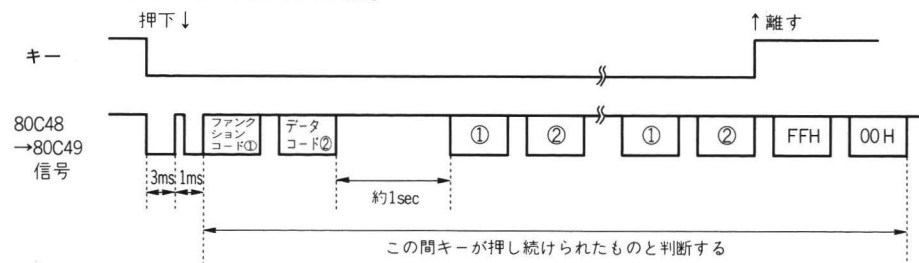
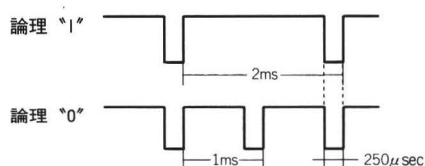
上位＼C	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		ᶜP	(SPACE)	0	@	P	`	p	▁	─		ー	タ	ミ	●	▒
1	ᶜA	ᶜQ	!	1	A	Q	a	q	▂	│	。	ア	チ	ム	○	土
2	ᶜB	ᶜR or INS	"	2	B	R	b	r	▃	┴	「	イ	ツ	メ	♠	金
3	ᶜC or BREAK	ᶜS	#	3	C	S	c	s	▄	┬	」	ウ	テ	モ	◆	木
4	ᶜD	ᶜT	$	4	D	T	d	t	▅	┤	、	エ	ト	ヤ	♥	水
5	ᶜE	ᶜU	%	5	E	U	e	u	▆	├	・	オ	ナ	ユ	♣	火
6	ᶜF	ᶜV	&	6	F	V	f	v	▇	┼	ヲ	カ	ニ	ヨ	◢	月
7	ᶜG	ᶜW	'	7	G	W	g	w	█	┐	ァ	キ	ヌ	ラ	◣	日
8	ᶜH or DEL	ᶜX	(	8	H	X	h	x	▏	┘	ィ	ク	ネ	リ	×	時
9	ᶜI or H TABE	ᶜY	)	9	I	Y	i	y	▎	└	ゥ	ケ	ノ	ル	▘	分
A	ᶜJ	ᶜZ	*	:	J	Z	j	z	▍	┌	ェ	コ	ハ	レ	▖	秒
B	ᶜK or HOME	ESC	+	;	K	[	k	{	▌	╮	ォ	サ	ヒ	ロ	▝	年
C	ᶜL or CLR	→	,	<	L	¥	l	|	▋	╰	ャ	シ	フ	ワ	▗	円
D	ᶜM or CR	←	−	=	M	]	m	}	▊	╯	ュ	ス	ヘ	ン	▞	人
E	ᶜN	↑	.	>	N	∧	n	─	▉	╭	ョ	セ	ホ	゛	▚	生
F	ᶜO	↓	/	?	O	_	o	π	█	╲	ッ	ソ	マ	゜	⊓	〒

図III-48 80C48出力コード表(テンキー,ファンクションキー,TVキー,カセットキー)

Low＼High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0						$G_=$			EJECT			
1				1				F1		$G_.$			STOP		VOL UP	
2				2				F2		G_2					VOL DOWN	
3	BREAK			3				F3		G_8			FF			
4				4				F4		G_6			REW			
5				5				F5		G_4			S_{FF}		T/C	
6				6				S_{F1}		G_5			S_{REW}			
7				7				S_{F2}		G_9						
8				8				S_{F3}		G_3						
9				9				S_{F4}		G_1						
A			*					S_{F5}		G_7						
B	HOME		+							G_*					CH UP	
C	CLR	→	,							G_-					CH DOWN	
D	CR	←	−	=						G_+						
E		↑	.							$G_/$						
F		↓	/							$G_,$						

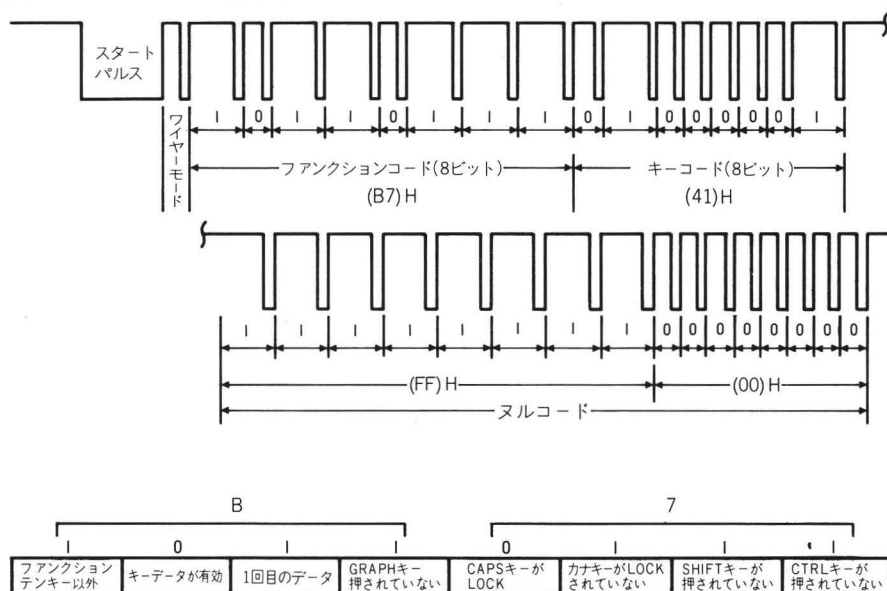
S：SHIFTキーを押しながら　　G：GRAPHキーを押しながら

伝送方式はPPM(パルス位置変調方式)で16ビット分が1回で送られます。そして最後に，必ずFF00Hが送出されます(図III-49)。

図III-49　80C48→80C49信号

たとえば，CAPS LOCKをし[A]キー(ASCIIコード41H)を打つとキーデータ出力波形は図III-50のようになります。ファンクションコードがB7Hということは，図III-46のビット構成と図III-50の下の説明により理解することができます。

リアルタイムキースキャンとロールオーバーの判定

ロールオーバーとは，[A]キーを押しながら[B]キーを押したような場合，ABと表示される

図III-50　キーデータ構成図例

機能をいいます。少し速いタイピングができるようになると必要になってくる機能です。比較的高級なキー処理方法に属します。

X1 のロールオーバー機能は，同時に 3 キー以上のロールオーバーも判断することができ，キーボード単体では最高の水準にあります。

X1 では，わずか 1 本の信号線で，ロールオーバーとリアルタイムキースキャンとを実現するため，80C48 と 80C49 の間に綿密な通信手順が決められています。

キー入力があると 80C48 からはそのキーに応じたファンクションコードと，ASCII コードの 2 バイトが続けて送られてきます。

キーが離されると，すぐに FF00H が送られますが，キーがそのまま押し続けられると FF00H は送出されず，約 1 秒後にリピートデータが繰り返し送られます。これは，キーが離されるまで続き，最後に FF00H が送出されます(図III-51)。

ロールオーバーの判定は，[A]キーの押し下げ中にBキーの押し下げがあった場合，その瞬間から[B]キーが有効になります。ふたつ以上のキーが同時に押されたとき，わずかな時間差によって，先に押されたキーのデータが必ず 1 回送出され，それ以後はあとに押されたキーのデータが有効になるのです。

ふたつ以上のキーの同時入力を判断するには割り込みによるキー入力処理が適しています。

図III-51　キースキャンとロールオーバーの判定

ロールオーバーの判定

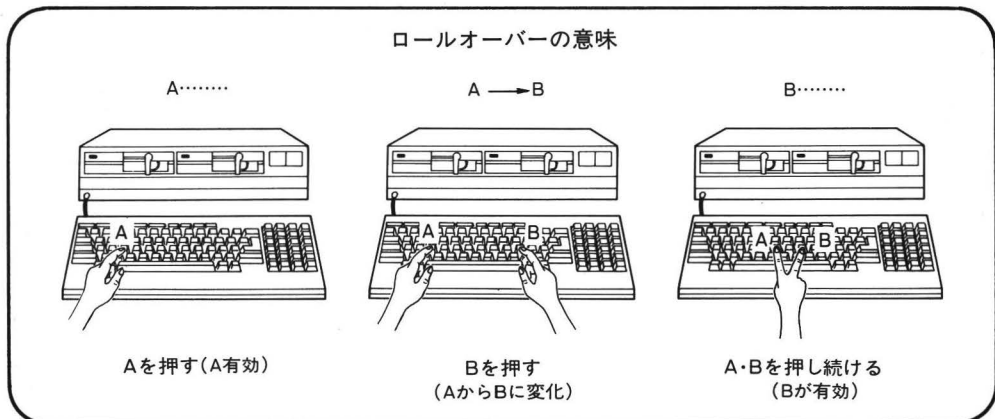
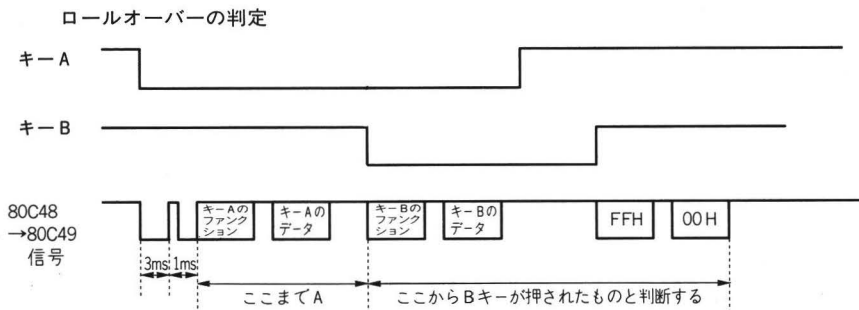
キーA

キーB

80C48
→80C49
信号

キーAのファンクション　キーAのデータ　キーBのファンクション　キーBのデータ　FFH　00H

3ms　1ms

ここまでA　ここからBキーが押されたものと判断する

X1 turboの特別機能

ゲームモード

X1 turboではキーボード横のスイッチを切り換えることにより，複数キー入力処理がスムーズに実行できるモードが用意されています。

ゲームキーデータは24ビットで構成されており，図III-52のようにそれぞれのビットが1個のキーに対応しています。

図III-52　X I turboゲームキーデータ

ビット	1	2	3	4	5	6	7	8
キー	Q	W	E	A	D	Z	X	C

ビット	9	10	11	12	13	14	15	16
キー	7	4	1	8	2	9	6	3

（ビット9～16：テンキー）

ビット	17	18	19	20	21	22	23	24
キー	ESC	／	－	＋	*	HTAB	スペース	↵

（ビット18～21：テンキー）

ゲームキーデータは図III-52のうち押されたキーのみに対応するビットを1とした3バイト信号となります(図III-53)。

図III-53　ゲームキーデータ（3バイト信号）

スイッチがゲームモードになっていても，先に説明した2バイト(キーデータ)は送られてくることに注意してください。ゲームモードになっているときのキーデータの構成は図III-54のとおりです。

図III-54　ゲームモード　キーデータ構成

①ゲームキーを押した(離した)とき

②リピート時またはゲームキー以外を押したとき

ここで述べているデータ構成はあくまでもキーボードと本体内の80C49との交信時のものであり，ユーザーがプログラムする場合はスタートパルスについては考える必要はあ

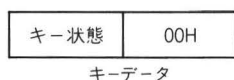
りません。ゲームキーを読み込めというコード(E3H)と得られたデータを送れというコード(E6H)をZ80CPUから80C49に送ればよいのです。具体的な方法はキー入力処理の方法のところで説明します。

特殊キー入力

X1 turboでは，GRAPH，カナ，CTRLのようなキーの入力も検出が可能となっています。特殊キーを押したときあるいは離したときのみ，図III-55に示すように2バイトのキーデータが送られます。キーコードバイトが00Hというところに注意してください。

図III-55　X1 turbo特殊キー入力

キー状態	00H

キーデータ

ファンクションコードの内容は図III-56のとおりです。

図III-56　X1 turboファンクションコード内容

			GRAPH	CAPS	カナ	SHIFT	CTRL
1	1	1	X	X	X	X	X

図中の×はそれぞれのキーが押されたなら1，そうでないのなら0となります。

μPD1990(タイマーカレンダ)

μPD1990は，時刻とカレンダのカウンタを内蔵しています。

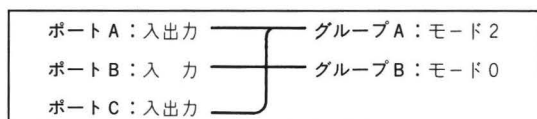
データの設定および読み出しは，シリアルに行われます。80C49は，メインCPUから送られる並列データをシリアルに変換してセットしたり，読み出したデータを並列に変換してメインCPUに送ります。またこのデータを利用して8チャンネルのタイマーの制御を行います。

このICはX1システムの電源がOFFになっても，ニッカド電池でバックアップされています。またシリアル入出力端子は，データのリード/ライト時以外は1Hzを出力しています。X1はこの信号をキャラクタのブリンクに使うBLINKCLOCKとして利用しています。

8255①

X1では，ふたつの8255(パラレルインタフェースIC)を使用しています。このうちのひとつ8225①は80C49の管理下にあって，80C49から図III-57のように初期設定されています。

サブCPUからメインCPUへの割り込みベクタは，ハード的に8255では管理できないのでサブCPU80C49が管理することになります。

図III-57　8255①の入出力モード

■メインCPUとサブCPUの交信

メインCPUとサブCPUの接続関係

Z80Aは，図III-58に示すようにふたつの8255プログラマブルインタフェースICを通して，カセットやテレビコントロールのデータを80C49へ送ったり，タイマーやカセットの状態を受け取ったりします。

このうち8255①のデータバスは，80C49のデータバスに接続されていて，初期設定などのコントロールを，Z80Aではなく80C49側から行います。そして，8255①のポートC第5，第7ビットからは，会話のタイミング合わせに重要なふたつの信号――IBF, OBFが出ています。

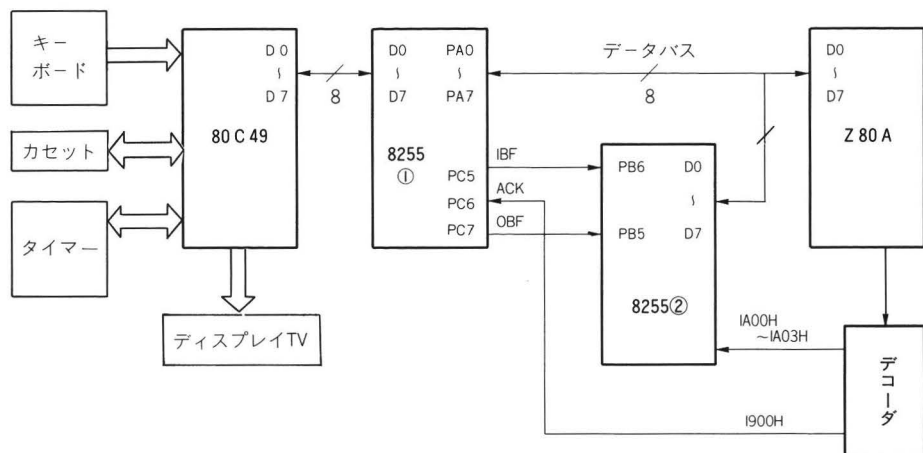
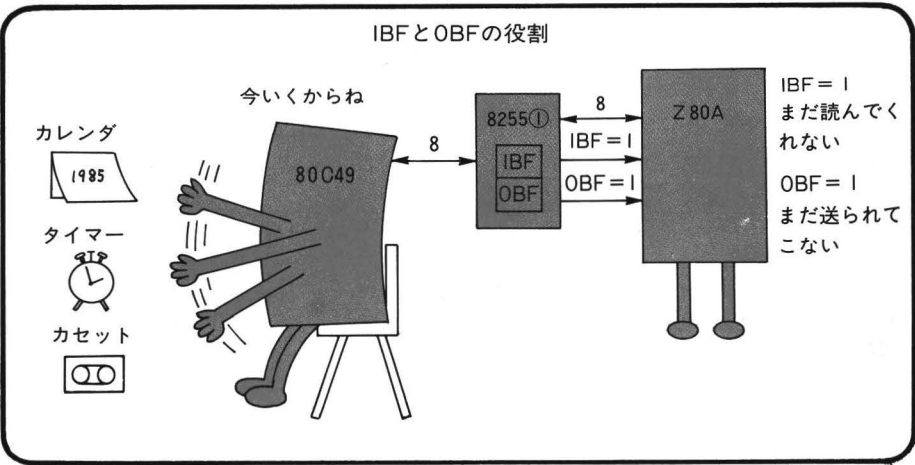
IBFは8255①がデータを受けつける状態にあるかどうかを判断する信号で，これが0にならないとZ80Aから80C49にデータを送ってはいけないことになります。IBFが1のときは0になるまで次のデータを送るのを待ちます。

一方のOBFは，80C49から8255①にデータが送られているかどうかを判断する信号です。これが1であれば80C49からデータが送られてきておりZ80Aはデータを読んでよいことになります。またOBFはZ80Aがデータを読み出すと0になります。0である間は，80C49からデータがまだ送られていないことを示し，読み出す必要のあるときはこれが1になるまで待ちます。

このIBFとOBFは，それぞれ8255②のポートB第6，第5ビットに接続されているのでZ80Aは，8255②を通してIBFとOBFを読み出すことができます。

ソフトを組むときはBCレジスタに1A01Hをセットし，IN命令で8255②のポートBの内容を見て判断します。

I/Oポートのアドレスと各ポートの働きを図III-59にまとめました。

図III-58　Z80A－80C49間の通信と制御

図III-59 8255①,②制御I/Oポート

I/Oアドレス		内容		アクティブ	初期設定
1900H	8255①ポートA	80C49との交信データ			モード2(双方向)
1A00H	8255②ポートA	プリンターデータ			モード1(出力)
1A01H	8255②ポートB	PB7	垂直帰線期間信号（入力）	L	モード0(入力)
		PB6	8255①のIBF信号（入力）	H	
		PB5	〃 OBF （入力）	L	
		PB4			
		PB3	プリンターのBUSY信号（入力）	L	
		PB2	垂直同期信号（入力）	H	
		PB1	カセット読み出しデータ（入力）		
		PB0	BREAK信号（入力）	L	
1A02H	8255②ポートC	PC7	プリンタのSTROBE（出力）	↑	モード1(出力)
		PC6	80/40文字モード切り換え（出力）	H(40)	
		PC5	G RAMアクセスモード（出力）	H (同時アクセス)	
		PC4	スムーズスクロール信号（出力）	L	
		PC3			
		PC2			
		PC1			
		PC0	カセットテープの書き込みデータ(出力)		
1A03H		モード設定			

1バイトデータの送受信

実際にZ80Aから80C49に対して1バイトのデータを送ったり受けとる場合は次のようにします。

① Z80Aから80C49に1バイトのデータを送る場合(Accに送りたいデータをセットする・リストIII-17)。

リストIII-17

Z80Aから80C49への1バイト送信ルーチン (Accに送るデータを設定)

```
          LD      A,DATA       : Accに送りたいデータをセット
TRNS49:   PUSH    AF
          LD      BC,1A01H     : 8255②のポートBのアドレス1A01HをBCレジスタに設定する
IBFCHECK: IN      A,(C)        : 8255②のポートBの内容をAccに取り込むIBF
          AND     40H          : IBFをチェックする(第6ビット)
          JR      NZ,IBFCHECK  : IBF="0"/"1"の場合 (下記参照)
          LD      BC,1900H     : 8255①のポートAのアドレス1900HをBCレジスタに設定する
          POP     AF           : Accに80C49に送るデータをセット
          OUT     (C),A        : OUT命令により，Accの内容を80C49に送る
          RET
```

IBF="0"の場合　8255①の入力バッファは空であり，80C49にデータを送ってもよい
IBF="1"の場合　8255①の入力バッファにデータが入っており，80C49がそのデータを読み込むまでZ80Aはデータを送るのを待つ必要がある

注）これと同じ働きをするルーチンはIOCSの0B54H，IPL ROMの052EHに存在する。実際に利用する際は会話のタイミングが狂わないように割り込み禁止をしてからCALLする

② Z80Aが80C49から1バイトのデータを受け取る場合(Accにデータを受ける・リストIII-18)。

リストIII-18

Z80Aが80C49から1バイトデータ受信ルーチン (Accにデータを受けとる)

```
RECV49:   LD      BC,1A01H     : 8255②のポートBのアドレス1A01HをBCレジスタに設定
OBFCHECK: IN      A,(C)        : 8255②のポートBの内容をAccに取り込む　OBF…ビット5
          AND     20H          : OBFをチェックする(第5ビット)
          JR      NZ,OBFCHECK  : OBF="0"/"1"の場合 (下記参照)
          LD      BC,1900H     : 8255①のポートAのアドレス1900HをBCレジスタに設定
          IN      A,(C)        : IN命令によりAccにデータを取り込む
          RET
```

OBF="0"の場合　8255①の出力バッファは空であり，80C49からのデータは送られていない
OBF="1"の場合　8255①の出力バッファに80C49からのデータが送られてきており，80C49側はZ80Aがデータの受け取りを待っている

注）X1ではこれと同じ働きをするルーチンはIOCSの0B49H，IPLROMの0523Hに存在する。実際に利用する際は会話のタイミングが狂わないように割り込み禁止をしてからCALLする

ここで紹介した80C49と1バイトのデータをやりとりするルーチン(TRNS49とRECV49)はとても重要なものです。以後のサンプルプログラムで，

```
CALL      TRNS49
```

などと，出てきたらこのルーチンを意味していることに注意してください。

メインCPUとサブCPUの会話方法

Z80A(メインCPU)と80C49(サブCPU)は，データ交換するために一定の会話手続きを

決めています。

Z80Aが80C49に何かをやらせたり，情報を得たりするときは，Z80Aから先に1バイトの送信要求コードが送られます。これはE4HからDFHまでのいずれかの数字に決められていて，80C49はこれを受けて次のことを判断します。

1．処理内容

2．データの方向

3．後続データのバイト数

送信要求コードの意味，方向，データのバイト数を図III-60にまとめておきます。

いま仮にZ80AからE9Hのコードが送られてきたとすると，80C49はカセットのコントロールと判断し，次の1バイトのデータが送られてくるのを待ちます。次に送られてきたコードが04Hであれば，巻き戻しが実行されます。

また最初にEAHが送られてきたとすると，80C49はカセットの状態を検出して，Z80Aにそのデータを送るので，Z80Aはすぐに受信の準備をしなければなりません。

このようにしてZ80Aと80C49は受信要求コードを通して会話を行います。

会話の最中に割り込みがかかってタイミングが狂わないように割り込みを禁止してください。

図III-60　送信要求コード表

送信要求コード	内容	後続バイト数	方向 80C49 Z-80A	説明
E3	ゲームキーデータ要求	3	→	ゲームキー入力データ(3バイト)を要求する(X1 turboだけの機能)
E4	Keyベクタ値をセット	1	←	Z 80A keyのベクタアドレスのロウバイトを返す ただし，0の場合にはkey割り込み禁止モードとなる
E6	Keyバッファ読み出し	2	←	80C49のkeyバッファの内容をZ80Aへ送る keyバッファには最初のデータが格納されている
E7	TV送信コードセット	1	←	Z 80Aより送信されてきたコードを80C49のP27端子よりTVへ送る
E8	TV送信コード読み出し	1	→	TVに最後に送られたコードをZ80Aへ返す
E9	カセット指示	1	←	カセットメカの動作をする
EA	カセット状態読み出し	1	→	カセットメカの動作状態を読み出しZ 80Aへ返す
EB	カセットセンサ読み出し	1	→	カセットセンサーを読み，その状態をZ 80Aへ返す
EC	日付けセット	3	←	時計用ICに"月"，"日"，"曜日"のデータを書き込む
ED	日付け読み出し	3	→	時計用ICから"月"，"日"，"曜日"のデータを読み出してZ80Aへ返す
EE	時刻セット	3	←	時計用ICに"時"，"分"，"秒"のデータを書き込む
EF	時刻読み出し	3	→	時計用ICから"時"，"分"，"秒"のデータを読み出す
D0 〜 D7	タイマー0(1,2,3,4,5,6,7)をセット(計8個)	6	←	タイマー0(1,2,3,4,5,6,7)の領域にデータを設定する
D8 〜 DF	タイマー0(1,2,3,4,5,6,7)を読み出す	6	→	タイマー0(1,2,3,4,5,6,7)の領域からデータを読み出す

サブCPUのコントロール機能

■キー入力処理の方法

送信要求コード

キーベクタ値のセット(E4H)
キーバッファ読み出し(E6H)
ゲームキー要求(E3H/X1 turbo)

X1のキー入力処理の最大の特徴は，割り込みを用いていることですが，割り込みを用いないでキー入力処理をすることもでき，どちらにするかはZ80A側で自由に選択できます。

割り込みは，外部機器からのハードウェア信号がCPUに伝わった時点で，その入出力処理ルーチンにとび実行するものです。割り込みによらないプログラムでCPUが外部機能の状態をチェックするのに比べてスピードの面からも，ソフトウェアの負担からも有効といえます。

このふたつのキー入力処理の方法を説明します。

割り込みによるキー入力処理

割り込みによるキー入力処理には，Z80Aの3種類の割り込みモード(IM・インタラプトモード=0,1,2)のうち割り込みモード2(IM=2)が使われています。

IM=0は，現在ではあまり使われていません。IM=1は割り込みモードとして広く用いられ，マシン語を扱う人の中には経験者も多いことでしょう。

これに対して，割り込みモード2(IM=2)は，Z80Aの割り込みモード中もっとも高度で拡張性が高いモードですが，経験した人が少ないと思われますので説明します。

IM=2の目的は，割り込み処理するためのルーチンをメインメモリ上の自由なアドレスに置けるようにすること，また多重割り込みを回路にしばられず実現することです。

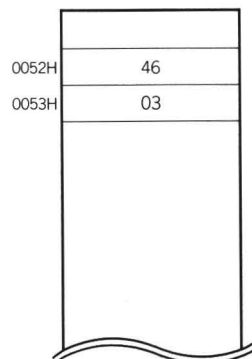
X1ではIOCSのキー入力割り込み処理ルーチンは0346Hにありますが，これを設定する場合を考えてみます。

まずベクタアドレスを決めます。ここでは0052Hとします。ベクタアドレスから2バイトに割り込み処理ルーチンのアドレスを設定します。L，Hの順で書き込みます(ベクタアドレスは必ず偶数でなければなりません・図III-61)。

ベクタアドレスの上位8ビット(00H)は，Z80AのIレジスタにセットします。

そして，下位8ビット(52H)は，割り込みをかける周辺機器に覚えさせておきます。こう

図III-61　モード2割り込みアドレス

アドレス	内容
0052H	46
0053H	03

すると Z80A に周辺機器から割り込みがかかると，割り込み禁止でなければ Z80A から $\overline{\text{IORQ}}$ 信号が送られます。周辺機器はこの信号を受けて，覚えていたベクタアドレス(下位)を Z80A に送ります。Z80A は I レジスタに覚えていたベクタアドレス(上位)と組み合わせて，ベクタアドレスの 0052H の内容を見ます。

ここに 0346H が書かれているので，0346H がサブルーチンコールされる——といった手順で割り込み処理が実行されます。

この方法を用いると，周辺機器が多くなっても周辺機器によってベクタを変えておけば，どの機器から割り込みがかかったかがすぐわかるようになります。割り込み処理ルーチンの中では，すべてのレジスタが保存されるように配慮してください。

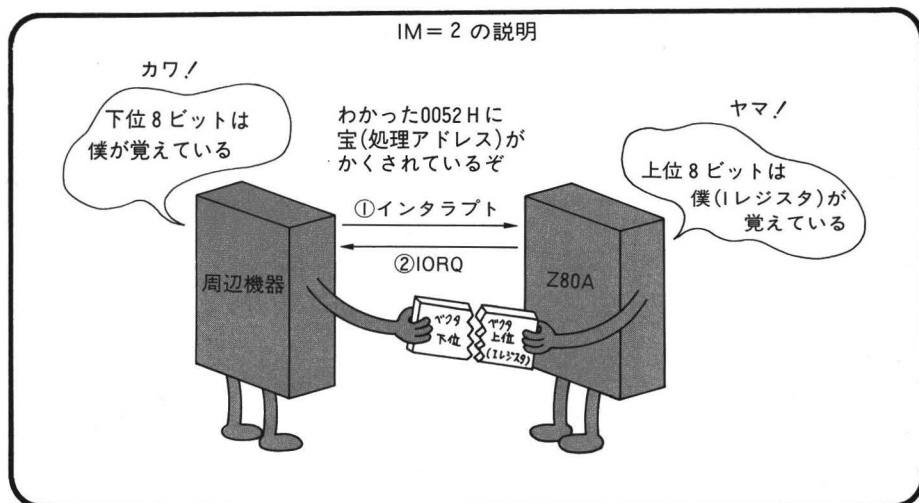
BASIC では，プログラム実行中でもしばしばブレークキーのチェックを行っていたため，処理速度が低下する大きな原因となっていました。しかし割り込みによるキー入力処理を用いると，このようにプログラム中にキー入力をチェックする必要がなくなるので実行速度が向上します。

さてベクタアドレスは，一般に周辺 IC(たとえば 8255 など)が管理しますが，キー処理の場合，8255 ①は 80C49 側からコントロールされる形をとっているため，80C49 が直接ベクタを管理しなければなりません。80C49 にベクタを知らせるにはコード E4H を使います。

実際にセットする場合はサブルーチン TRNS49 を用いて，Z80A から E4H を送り続けてベクタアドレス下位を送ります。送るベクタは 00H 以外の偶数でなければなりません。

X1 のシステムでは 00H は特別な意味を持ち，80C49 はこれが送られてくると割り込みによらないでキー入力処理をするものと判断します。

割り込みキー処理をセットするルーチンと，キー処理をするルーチンの一例をリストIII-19 に示します。VECT は割り込みベクタアドレスで図III-61 の 0052H に対応します。

ここで注意しなければならないのは，割り込みを禁止した場合です。割り込み禁止中に入力されたキーデータは，Z80A が読み出すまで 80C49 内のキーバッファに蓄えられます。蓄えられるのは最大8文字分(16バイト)のデータまでです(これは 80C49 内のキーバッファ

リストIII-19

キー入力割り込み処理セットルーチン（プログラムの最初に必ず一度セットする必要がある）

```
INITKEY: IM      2              ：割り込みモード２とする
         LD      HL,KEYIN
         LD      (VECT),HL
         LD      A,H
         LD      I,A            ：Ｉレジスタにベクタアドレス上位８ビットをセットする
         LD      A,0E4H       ]  80C49と同期をとりながら"E 4"を送る
         CALL    TRNS49
         LD      A,L          ]  ベクタアドレス下位８ビットを80C49へ知らせる
         CALL    TRNS49
;
;        (EXIT)
;
```

割り込みキー処理ルーチン（メインプログラムの流れとは別に作る必要がある）

```
KEYIN:   CALL    RECV49       ]  ファンクションコード部の読み込み
         LD      H,A
         CALL    RECV49       ]  ASC IIコード部の読み込み
         LD      L,A
;
;        (EXIT)                 ：HLレジスタにセットされたキーデータにより各処理を行う
;
```

の値です。CZ-8CB01内のIOCSにはこれと別にソフトによる64バイト・64文字分のキーバッファが設けられている)。それ以上入力されたキーは無視されます。割り込み禁止が解除されると，この8文字分のデータが一度に続けて80C49から送られるため，ソフト側で何らかの対策が必要な場合も出てきます。

割り込みによらないキー入力処理

この方法は，リアルタイムキー入力や先行入力バッファが不要なときに利用します。

80C49からのキー入力割り込みを止めるには，E4Hに続いて00Hを送ります。80C49はこれを受けると，割り込みによらない方法でキー入力処理するものと判断し，キーボードから入力があってもZ80Aに割り込みをかけません。このためキー入力を見るためにはZ80Aから80C49に働きかけてキー入力データを要求する必要があります。

80C49からキー入力データを受け取るには，まずZ80Aから80C49にコードE6Hを送

リストIII-20

割り込みによらないキー処理例

```
         LD      A,0E4H       ]  80C49と同期をとり"E 4"Hを転送する
         CALL    TRNS49
         LD      A,00H        ]  80C49へ"0"を送り80C49からの割り込みキー入力を禁止する
         CALL    TRNS49
         LD      A,0E6H       ]  80C49に"E 6H"を送りキーバッファの内容を要求する
         CALL    TRNS49
         CALL    RECV49       ]  キーのファンクションコード部をHレジスタにロードする
         LD      H,A
         CALL    RECV49       ]  キーのキーコード部をLレジスタにロードする
         LD      L,A
;
;        (EXIT)               ]  キー入力に応じた処理をする
;
```

ります。すると80C49から2バイトのキーデータが，ファンクションコード部，キーコード部の順で送られてきます。このデータは最後にキーボードの80C48/49から80C49へ送られてきたデータなので，リアルタイムキースキャンに近いものとなります。この場合の例(リストIII-20)を紹介します。

ただしモニタから操作する場合は，戻る前にベクタアドレスをリストIII-21のように設定し直すことを忘れないでください。

```
リストIII-21
LD      A,0E4H
CALL    TRNS49
LD      A,52H
CALL    TRNS49
RET
```

なおブレークキーが押された場合の処理は普通のキー処理とまったく同じ(キーコード03H)ですが，このとき同時に8255②ポートBの第0ビットが0になりますので，これを見てブレークの有無を知ることもできます。

またカセットがREADかWRITE状態にあるときカセットキーが押されるとブレークキーが押されたときと同じ動作をします。

■画面モードとテレビのコントロール

制御の意味

送信要求コード(E7H)

X1は，専用モニタを用いることによって，コンピュータから画面モード(テレビ，コンピュータ画面の選択)，チャンネル選択，ボリューム調節，パワーのON/OFFの制御ができます。制御コードの送出は80C49を通して行われますが，Z80Aから画面を制御するためには80C49にコードE7Hに続けて制御コードを送ります。

では実際の制御方法をみてみましょう。

画面モードの制御方法

専用モニタは，次の4種類の画面モードをとることができます。

1．テレビ画面(CRT＝0)

2．コンピュータ画面(CRT＝1)

3．スーパーインポーズをさせてコントラストダウン(CRT＝2)

4．スーパーインポーズをさせてコントラストは不変化(CRT＝3)

これらのモードを制御するには，送信要求コードに続いて1バイトの制御コードを送ります。このとき，モードによっては1～3回制御コードを送ります。

これをまとめたのが図III-62です。プログラムにすると，リストIII-22のようになります。

チャンネル，ボリュウム，パワーのON/OFF制御

専用モニタのチャンネル，ボリューム，パワーのON/OFFも，コンピュータから制御で

図III-62　テレビ，コンピュータ画面制御送信コード表

画面モード ＼ 送信コード(バイト数)	1	2	3	4	5	6
TV画面	E7	05				
コンピュータ画面	E7	05	E7	08		
スーパーインポーズ 1（コントラスト ダウン）	E7	05	E7	0F	E7	0A
スーパーインポーズ 2（コントラスト ノーマル）	E7	05	E7	0F		

きます。

チャンネルの選択は，直接1～12の任意のチャンネルを指定することも，アップ/ダウンで切り変えることもできます。

ボリュームは64段階に変えることができ，アップ/ダウンで変えることも，ミュートや標準音量にすることもできます。またパワーのON/OFFはトグルで換える(ONならOFF，OFFならON)こともできるし，直接ON/OFFすることもできます。これらの制御はE7Hに続いて図III-63に示す制御コードを送ることによって行うことができます。たとえば，パワーONの制御はリストIII-23のようにします。

テレビに最後に送られたコードの読み出し

送信要求コード(E8H)

このコードを用いることによって，最後にテレビに送られたコードを読み出すことができます。コードE8Hを80C49に送ると，80C49から1バイトの制御コード(最後に送られたコード)が返されます。次のプログラムは，最後に送られたコードを読み出し，表示するプログラム(リストIII-24)の1例です。

図III-63　テレビ制御送信コード表

内容 ＼ 送信コード(バイト数)	1	2
ボリュームアップ	E7	01
ボリュームダウン	E7	02
ボリュームノーマル(42/64階調)	E7	03
音声ミュート	E7	06
チャンネルアップ	E7	0B
チャンネルダウン	E7	0C
パワーオフ	E7	0D
パワーオン/オフ(トグル動作)	E7	0E
チャンネル 1 ～ チャンネル 12	E7	10 ～ 1B
パワーオン	E7	80

リストIII-23

```
          LD        A,0E7H
          CALL      TRNS49
          LD        A,80H
          CALL      TRNS49
;
;         (EXIT)
;
```

リストIII-24

```
ACCHEX    EQU       1207H

          LD        A,0E8H
          CALL      TRNS49
          CALL      RECV49
          CALL      ACCHEX
;
;         (EXIT)
;
```

リストIII-22

TV画面設定例

```
TVDISP: LD      A,0E7H
        CALL    TRNS49
        LD      A,05H
        CALL    TRNS49
;
;       (EXIT)
;
```

コンピュータ画面設定例

```
COMDISP:LD      A,0E7H
        CALL    TRNS49
        LD      A,05H
        CALL    TRNS49
        LD      A,0E7H
        CALL    TRNS49
        LD      A,08H
        CALL    TRNS49
;
;       (EXIT)
;
```

スーパーインポーズ1モード設定例（TVコントラストダウン）

```
SUPER1: LD      B,06H       :送信バイト数を指定する
        LD      DE,TVDAT1
SPRLP1: LD      A,(DE)
        CALL    TRNS49
        INC     DE
        DJNZ    SPRLP1
;
;       (EXIT)
;
TVDAT1: DB      0E7H
        DB      05H
        DB      0E7H
        DB      0FH
        DB      0E7H
        DB      0AH
```

スーパーインポーズ2モード設定例（TVコントラストノーマル）

```
SUPER2: LD      B,04H       :送信バイト数を指定する
        LD      DE,TVDAT2
SPRLP2: LD      A,(DE)
        CALL    TRNS49
        INC     DE
        DJNZ    SPRLP2
;
;       (EXIT)
;
TVDAT2: DB      0E7H
        DB      05H
        DB      0E7H
        DB      0FH
```

■時刻の設定と読み出し

設定と読み出し法

送信要求コード

日付の設定(ECH)，読み出し(EDH)

時刻の設定(EEH)，読み出し(EFH)

カレンダ時計μPD1990は，80C49の管理下にあります。両者の間のデータ交換は直列に行われますが，直並列変換と並直列変換はすべて80C49が行うため，Z80Aは80C49に対し，並列データを送受信するだけですみます。μPD1990は時計機能をコントロールします。ただし“年”のカウントだけは80C49が行います。日付と時刻はそれぞれ単独に設定，読み出しができます(図III-64)。Z80Aと80C49との間で交換されるデータは3バイト構成で，図III－65のようになっています。なお，ここで使われる数字の表現は“月”を除いてBCD(Binary Coded Decimal：2進化10進数)となっています。

図III-64　日付け，時刻データ設定，読み出し送信コード表

a）設定(80C49-Z-80A)

内容	送信コード
日付	″EC″＋3バイトデータ
時刻	″EE″＋3バイトデータ

b）読み出し(80C49-Z-80A)

内容	送受信コード
日付	″ED″＋3バイトデータ
時刻	″EF″＋3バイトデータ

日付，時刻の設定例

日付：'83年11月23日水曜日

時刻：　14時　56分　7秒

設定例をリストIII-25に示します。

リストIII-25

```
CLND:    LD      B,08H
         LD      DE,CLNDAT
CLNDLP:  LD      A,(DE)
         CALL    TRNS49
         INC     DE            80C49に順次コードおよびデータを送る
         DJNZ    CLNDLP
;
;        (EXIT)
;
CLNDAT:  DB      0ECH          ：日付設定送信コード
         DB      83H           ：'83年
         DB      0B3H          ：11月，水曜日
         DB      23H           ：23日
         DB      0EEH          ：時刻設定送信コード
         DB      14H           ：14時
         DB      56H           ：56分
         DB      07H           ：7秒
```

図Ⅲ-65　日付，時刻データ構成

項目	内容
年	ビット 7 6 5 4 3 2 1 0 10の位（ビット7〜4）　1の位（ビット3〜0） 00年〜99年の値をそのまま指定
月，曜日	ビット 7 6 5 4 3 2 1 0 月（ビット7〜4）　＊（ビット3）　曜日（ビット2〜0）　＊印は無効ビット 1H(月)〜CH(12月)の値を指定　0(日)〜6(土)の値を指定 曜日：0日→6土
日	ビット 7 6 5 4 3 2 1 0 10の位（ビット7〜4）　1の位（ビット3〜0） 10の位：0〜3Hまでの値を指定 1の位：1〜9Hまでの値を指定
時	ビット 7 6 5 4 3 2 1 0 10の位（ビット7〜4）　1の位（ビット3〜0） 10の位：0〜2Hまで値を指定 1の位：0〜9Hまで値を指定
分	ビット 7 6 5 4 3 2 1 0 ＊（ビット7）　10の位（ビット6〜4）　1の位（ビット3〜0）　＊印は無効ビット 10の位：0〜5Hまでの値を指定 1の位：1〜9Hまでの値を指定
秒	ビット 7 6 5 4 3 2 1 0 ＊（ビット7）　10の位（ビット6〜4）　1の位（ビット3〜0）　＊印は無効ビット 分と同じ指定法

日付，時刻の読み出し例

次に日付，時刻をDATDAT番地以降に読み出す例をリストIII-26に示します。

リストIII-26

```
        LD      A,0EDH       ] 日付読み出しコード(ED)Hを80C49へ転送
        CALL    TRNS49       ]
        LD      B,03H        ]
        LD      DE,DATDAT    ]
DRD:    CALL    RECV49       ] 3バイトデータをDTR番地より順次ストアして行く
        LD      (DE),A       ]
        INC     DE           ]
        DJNZ    DRD          ]
        LD      A,0EFH       ] 時刻読み出しコード（EF）Hを80C49へ転送
        CALL    TRNS49       ]
        LD      B,03H        ]
        LD      DE,TIMDAT    ]
TRD:    CALL    RECV49       ] 3バイトデータをDTR＋3番地より順次ストアして行く
        LD      (DE),A       ]
        INC     DE           ]
        DJNZ    TRD          ]
;
;       (EXIT)
;
```

たとえば次のようなデータが得られたとすると(DATDAT＝A040Hとした場合)，

```
A040:84 33 24 13 40 21
```

このデータは84年3月24日WED13時40分21秒と読むことができます。

■テレビタイマーの設定と読み出し

テレビタイマーの機能

送信要求コード

タイマーの設定(D0H～D7H)

タイマーの読み出し(D8H～DFH)

X1にはもうひとつの魅力としてテレビタイマーがあります。

テレビタイマーはサブCPU80C49が管理し，メインCPUとは独立に8個までのタイマーが設定できます。タイマー機能のうち7個のタイマーと，テレビのON/OFF,チャンネルの設定はBASICのASKから行えますが，マシン語レベルで直接タイマーを操作することによって，次のような機能が新しく利用できるようになります。

1．インタバルタイマーが使える

インタバルタイマーは，設定時刻がくると一定の時間間隔で同じ動作を繰り返すことができる機能です。そのインタバル周期は1～59分まで選ぶことができます。具体的には1分ごとにチャンネルを変えていくことができるわけです。

2．タイマ割り込みが使える

設定時刻がきたら，サブCPUからメインCPUに対して割り込みをかけることができます。インタバルタイマーとあわせて,1分単位で割り込みを発生させることができます。

3．すべてのテレビ制御がタイマーによって設定できる。

ASKからはテレビパワーのON/OFFとチャンネル操作だけしか設定できませんが，マシン語レベルで操作すると，すべてのテレビ制御ができます。たとえばボリュ

ームのUP, DOWN, ミュートなどが可能です。

4. タイマー番号“0”が使える。

タイマー0はASKから設定できませんが，マシン語レベルの操作によって，タイマー0を含めて最大8つまでのタイマーが設定できます。

5. カセット制御が設定できる

タイマーによるテレビ制御のほかに，タイマーによるカセット制御も可能です。たとえば設定時刻がくると早送りを実行したり，巻き戻しを実行したりすることができます。ただし，メインCPUの電源がONの期間中しか実行できません。

では，タイマーの設定と読み出し法についてみていきましょう。

タイマーデータの構成

タイマーを設定する場合，まずZ80Aから1バイトのタイマー設定コードを送り，続いて6バイトのタイマーデータを送ります。

設定コードはD0HからD7Hまでの数字で，それぞれがタイマー番号0から7までの8つのタイマーに対応しています。

読み出しの場合，まずZ80Aから1バイトのタイマー読み出しコードを送ると，80C49から6バイトのタイマーデータが送られてきます。読み出しコードはD8HからDFHまでの数字で，それぞれがタイマー番号0から7までに対応しています。

このタイマー番号と，設定および読み出しコードの対応を図III-66の表に示します。

図III-66 タイマ設定，読み出し送信コード

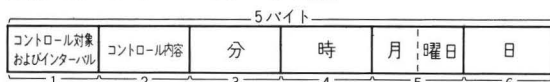
タイマ番号	設定コード	読み出しコード
0	D0	D8
1	D1	D9
2	D2	DA
3	D3	DB
4	D4	DC
5	D5	DD
6	D6	DE
7	D7	DF

この表のうち，タイマー番号0はASKからは操作できません。マシン語レベルの操作によってのみ利用できるタイマーです。

タイマーのデータは6バイトで，図III-67のような構成になっています。

図III-67 タイマーのデータ構成

コントロール対象およびインターバル	コントロール内容	分	時	月	曜日	日
1	2	3	4	5		6

(上部：5バイト)

コントロール対象およびインタバル(1バイト目)

第1バイトの8ビット構成(図III-68)のうち，ビット7とビット6によってタイマーの有効無効，対象がテレビかカセットか，割り込みタイマーか，ビット5～ビット0によってインタバルを設けるとすれば何分にするか――を決めています(図III-69)。

図III-68 コントロール対象とインタバル①

7	6	5	4	3	2	1	0
対象	有効無効	イ	ン	タ	バ	ル	1 ～ 59 分

インタバルタイマーは設定時間がくると，一定の時間(1～59分)が経過するごとに，2

バイト目(コントロール内容)のデータで指示された動作を繰り返す機能のことです。

たとえば一定の時間がくるたびに，音声をミュートしたり，割り込みタイマーをかけたり，カセットの巻きもどしをしたりすることができます。ただしカセットをコントロールする場合は，前面の電源がONのときでないと行えません。

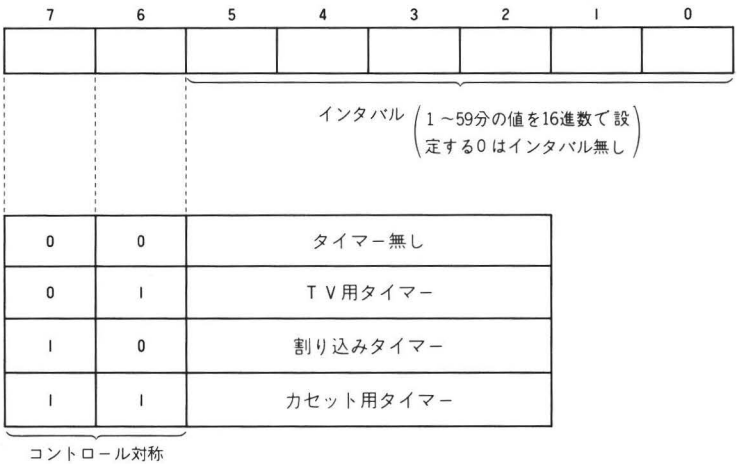
インタバルタイマーはBASICのASK命令からは設定することができなかった機能です。インタバルタイマーを0に設定すると無効になります。

図III-69　コントロール対象とインタバル②

7	6	5	4	3	2	1	0

ビット5〜0：インタバル（1〜59分の値を16進数で設定する0はインタバル無し）

7	6	
0	0	タイマー無し
0	1	TV用タイマー
1	0	割り込みタイマー
1	1	カセット用タイマー

ビット7・6：コントロール対称

コントロール内容(2バイト目)

このデータの内容はコントロール対象がテレビあるいは割り込みタイマーか，カセットかによって意味が異なってきます。

それぞれの場合についてみていきます。

① コントロール対象がテレビの場合

ビット7はパワーオンのコードをテレビに送るか否かを決めます。このビットが1のとき，パワーオンのコードがテレビに送られ，続いてビット4〜0で示される命令コードがテレビに送られます。

ビット6,5は常に0にセットします。

ビット4〜0の内容にしたがい，テレビに対して次に示すコントロールが行われます。図III-70には最上位ビット(bit7)が0の場合のコードを示します。

図III-70　TV用タイマーがコントロール対象の場合

コントロール内容	コード
タイマー無効	00
ボリュームアップ	01
ボリュームダウン	02
ボリュームノーマル	03
音声ミュート	06
チャンネルアップ	0B
チャンネルダウン	0C
パワーオフ	0D
パワーオン／オフ反転	0E
チャンネル1 ∫ チャンネル12	10 ∫ 1B

② コントロール対象が割り込みタイマーの場合

割り込みベクタの下位8ビットが設定されます。この機能はZ80が割り込みモードIM＝2に設定されていないと使えません。

タイマー割り込みは今のところCZ-8CB01およびCZ-8FB01ではサポートされていない機能です。

インタバルタイマーと割り込みタイマーを利用することによって，プログラム実行中の強制ブレーク，ON TIME，GO SUBに相当する機能が新しく使えるようになります。

③ コントロール対象がカセットの場合

カセットのコントロールコードが入ります。内容は図III-71のとおりです。

図III-71 カセットコントロールコード

コントロール内容	コード
EJECT	00
STOP	01
READ	02
FF	03
REW	04
APSS 1	05
APSS −1	06
WRITE	0A

“分”の設定（3バイト目）

上位3ビット(bit6～4)と下位4ビット(bit3～0)にそれぞれ10の位，1の位をBCDの形式でデータを設定(図III-72)します，設定値は0～59です。

最上位ビット(bit7)が1の場合，分の項は無効になります(マニアタイプのみ，それ以外はbit7は0にしてください)。

なおBCDとは2進化10進数のことで，たとえば45Hと設定された場合45分を示し69分ではありません。

図III-72 〝分〟データ設定値

“時”の設定（4バイト目）

上位4ビット(bit7～4)と下位4ビット(bit3～0)に，それぞれBCDの形式でデータを設定します。設定値は0～24です。FFHを設定すると，この項は無効となります。

“月，曜日”の設定（5バイト目）

上位4ビット(bit 7～4)と下位4ビット(bit3～0)に，それぞれ月，曜日のデータを設定します(図III-73)。“月”データの設定値は1～12の値を16進数で設定します。0を設定すると月のデータは無効になります。

“曜日”データの設定値は0～6で，日曜日～土曜日に対応しています。曜日データはFを設定すると無効になります。

図III-73 〝時〟および〝月，曜日〟データ設定値

図III-74 〝日〟データ設定値

“日”の設定（6バイト目）

上位4ビット（bit 7～4）と下位4ビット（bit 3～0）に，それぞれ10の位，1の位をBCDの形式でデータを設定します。設定値は1～31で0を設定すると無効となります（図III-74）。

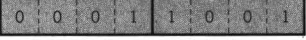
設定と読み出し例

テレビタイマーの設定

テレビタイマーの設定は，Z80Aからサブ CPUに対して1バイトの指示コードを送り，続いて6バイトのデータを送ることにより設定することができます。ここでは設定例（リストIII-27）として，通常，BASICのASK命令では設定できないインタバルタイマーを使用します。設定データは図III-75のとおりでタイマー番号1のタイマーに1分ごとのトグル動作（テレビパワーのON/OFFを繰り返す）を設定します。

リストIII-27

```
TIMSET:  LD      B,07H
         LD      DE,TIMDAT
TSETLP:  LD      A,(DE)
         CALL    TRNS49
         INC     DE
         DJNZ    TSETLP
         RET
;
TIMDAT:  DB      0D1H,41H,0EH,15H,0FFH,0FH,00H
```

テレビタイマーの読み出し

設定されたタイマーデータを読み出してみます。読み出しはZ80Aから，サブ CPU 80C49に読み出しコードを送った後，6バイトのコードをサブ CPUから受け取ります。

BASICのASK命令から，次のように，

04/18　THU　10:38 ON CH1

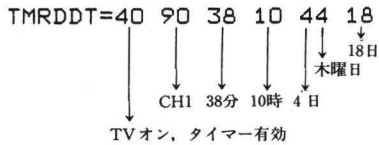
と設定します。これをリストIII-28で実行して読み出すと，読み出されたデータはTMRD DTに格納されます。

図III-75　TVタイマーの設定例

送信データ	データの内容	ビット
D1	タイマー番号/設定コード	
41	コントロール対称はTV，インターバルは1分毎	0 1 0 0 0 0 0 1
0E	コントロールの内容，トグル動作	0 0 0 0 1 1 1 0
15	15分（適当な時間を設定してください）	0 0 0 1 1 0 0 1
FF	時は無効	
0F	月，曜日は無効	
00	日は無効	

リストIII-28

```
        LD      A,0D9H
        CALL    TRNS49
        LD      B,06H
        LD      DE,TMRDDT
TIMRD:  CALL    RECV49
        LD      (DE),A
        INC     DE
        DJNZ    TIMRD
        RET
;
TMRDDT: DS      6
```

内容は次のとおりです。

割り込みタイマーの設定

割り込みタイマーは，設定した時刻がくるとサブCPUからZ80に対し割り込みをかける機能をいいます。割り込みタイマーは，インタバルタイマーと併用することにより，1〜59分間隔で割り込みを発生させることができます。割り込みはIM=2モードを用いますが，このモードの設定は少々むずかしいので注意が必要です。

リストIII-29

```
INTTAB  EQU     0054H


        LD      HL,BEEP1
        LD      (INTTAB),HL
        LD      B,07H
        LD      DE,INTTIM
ISETLP: LD      A,(DE)
        CALL    TRNS49
        INC     DE
        DJNZ    ISETLP
        RET
;
INTTIM: DB      0D0H,81H,54H,15H,10H,0FH,00H
;
BEEP1:  PUSH    AF
        PUSH    BC
        PUSH    DE
        PUSH    HL
        CALL    07F7H
        POP     HL
        POP     DE
        POP     BC
        POP     AF
        RET
```

割り込みタイマーの実際例として，設定時刻10時15分から，1分間隔でBEEP音を発生させるタイマーを設定してみます。IM=2の設定はHu-BASICからスタートすると設定されますので，BASICに任せることにして，まずBASICを起動してからモニタよりリストIII-29を実行してください。ここでは通常用いることのないタイマー0を使ってみます。

BASICではキー割り込みのベクタ値を0052Hに設定しています。割り込みベクタの上位8ビットはZ80のIレジスタに設定してあり，BASICでは00Hとなっていますが，ベクタ値の下位8ビット(52H)はあらかじめキー入力割り込みベクタとしてサブCPUに設定されています。このためキー割り込みがあると0052Hに格納されているアドレス0346Hから実行が始まります。

タイマー割り込みのベクタ値は0054Hとすることにした場合，0054Hから2バイトにBEEPを実行するサブルーチンのアドレスBEEP1を書き込みます。これによりタイマー割り込みがあるたびにBEEP1のサブルーチンが実行されます。

なお，この割り込みタイマーは前面の電源(Z80A側の電源)を切っても解除されませんので注意が必要です。解除するときは主電源を切るか，無効タイマーを設定します。

■カセットメカニズムの制御

カセットコントロール

送信要求コード

カセット制御(E9H)

カセットメカニズムは80C49の管理下にあるので，これをZ80Aから制御するには送信要求コードE9Hに続けて1バイトの制御コードを送ります。その制御コードは図III-76のとおりです。

また，リストIII-30(カセットEJECTプログラム)はそのサンプルプログラムです。

図III-76　カセット制御送信コード

動作 ＼ 送信コード(バイト数)	1	2
EJECT	E9	00
STOP	E9	01
PLAY	E9	02
FF	E9	03
REW	E9	04
APSS FF	E9	05
APSS REW	E9	06
REC	E9	0A

リストIII-30

カセットEJECTプログラム

```
        LD      A,0E9H
        CALL    TRNS49
        LD      A,00H    ：制御コード
        CALL    TRNS49
;
;       (EXIT)
;
```

カセット制御では，次のような点に注意が必要です。

APSS実行中やそのほかのメカニズム操作の直後は，80C49がコマンドを受け付けない状態になります。そのようなときにTRNS49またはRECV49をCALLすると，OBF(IBF)CHEKをまわり続けて，プログラムの実行はカセット動作が終わるまで停止します。

Hu-BASICではこれを利用してAPSS動作を行っていますが，IOCSを用いてリアルタイムキースキャンを行う場合などに不都合な場合もあります。そのようなときはOBF(IBF)CHECKにループカウンタをセットして脱出をはかるなどの工夫をしてください。

カセットメカニズムの状態の検出

送信要求コード

カセット状態読み出し(EAH)

カセットセンサー読み出し(EBH)

カセットメカニズムは80C49の管理下にあります。80C49はメカニズムの状態，テープエンド，カセットの有無，消去防止ツメの有無を常に監視し処理を行っています。

Z80Aがそれらの状態を読み出すには，80C49にEAH,EBHのコードを送ります。コードEAHはカセットのメカニズムの状態を問い合わせるコードで，80C49から図III-71と同じカセットの状態に応じた1バイトのコードが送られてきます。

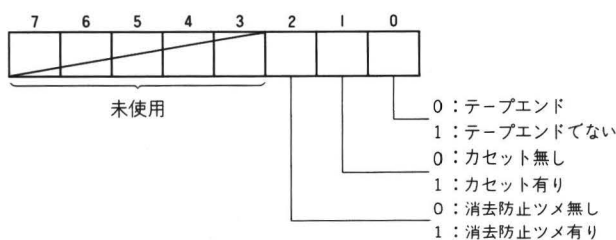
コードEBHを送ると，テープエンド，カセットの有無，消去防止ツメの有無の状態が各ビットに割り当てられて帰ってきます。各ビットに割り当てられた内容は図III-77のとおりです。

ところで，PLAYまたはREC実行中にブレーク(SHIFT＋BREAK)やカセットキーが押された場合は，次のようになります。

BREAKキーが押された時

a．カセットメカはストップ状態になります。

b．Z80Aに対して，BREAKのポート(8255②ポートB0)を“L”にします(または8255①ポートC1が“L”となる)割り込みがかかり，このときのキーコードは03Hです。

図III-77　カセットセンサ・データのビット構成

カセットキーが押されたとき

a．カセットメカは，押されたキーの内容に従います。

b．Z80AにたいしてBREAKのポートを“L”にします。

c．割り込みがかかり，このときのキーコードは03Hです。

このような80C49の応答を十分考慮したうえでソフトウェアの設計をします。

サンプルプログラムをリストIII-31に示します。

リストIII-31

カセットメカ状態の読み出し

```
CMTRD:  LD      A,0EAH
        CALL    TRNS49
        CALL    RECV49 : Accにメカ状態が入る
;
;       (EXIT)
;
```

カセットセンサの読み出し

```
CMTSNS: LD      A,0EBH
        CALL    TRNS49
        CALL    RECV49 : Accのビット0，1，2にセンサ検出の内容が入る
;
;       (EXIT)
;
```

PSGのハイテク活用法

PSGの機能とレジスタ

■PSGとは

X1シリーズは，サウンド発生用ICとしてAY-3-8910を使用しています。このICは，次のような機能を備えています。

① 3チャンネルのトーンジェネレータ
② 1チャンネルのノイズジェネレータ
③ 1チャンネルのエンベロープジェネレータ
④ ふた組のI/Oポート

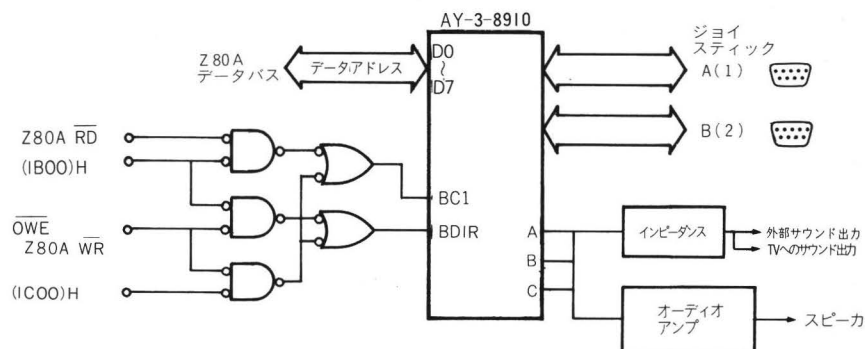
X1は，①，②により3重和音および雑音を実現し，さらに③により周期的波形を作ることができます。④のI/Oポートを2組のジョイスティック端子として利用しています。サウンドの発生はBASICのSOUNDコマンドで十分可能ですが，マシン語レベルで操作することによって，より多くの機能を引き出すことができるようになります。

サウンドICの各種の機能と，効果的な使い方についてみていきましょう。

サウンドIC (AY-3-8910)と周辺回路

サウンドIC周辺のブロック図を示します(図III-78)。サウンドIC内には16個のレジスタがあり図III-79に示すような働きをしています。

これらの各レジスタへのアクセスは，チップのふたつの端子BC1とBDIRの操作で行われます。アクセス方法は図III-78に示すように煩雑なため，Z80AからみてアクセスがBC1，BDIRに加えられます。

図III-78 AY-3-8910回りの信号系

図III-79　AY-3-8910 PSG内部レジスタの構成と機能

レジスタ	内容＼ビット	7	6	5	4	3	2	1	0
R_0	チャンネルA周波数	8ビット　(微　調　整)							
R_1						4ビット　(粗　調　整)			
R_2	チャンネルB周波数	8ビット　(微　調　整)							
R_3						4ビット　(粗　調　整)			
R_4	チャンネルC周波数	8ビット　(微　調　整)							
R_5						4ビット　(粗　調　整)			
R_6	ノイズ周波数				5ビット データ				
R_7	チャンネル設定	IN/OUT		ノイズ			トーン		
		IOB	IOA	C	B	A	C	B	A
R_8	チャンネルA音量				M	L3	L2	L1	L0
R_9	チャンネルB音量				M	L3	L2	L1	L0
R_{10}	チャンネルC音量				M	L3	L2	L1	L0
R_{11}	エンベロープ周期	8ビット FT							
R_{12}		8ビット CT							
R_{13}	エンベロープ形状					E3	E2	E1	E0
R_{14}	I/OポートAデータ	8ビット（パラレル）データ							
R_{15}	I/OポートBデータ	8ビット（パラレル）データ							

これによって，Z80A からは，

　レジスタの指定はI/O ポート 1C00H

　データの書き込み読み出しはI/O ポート 1B00H

をアクセスするだけですむように単純になっています(図III－80)。

たとえば，あるレジスタに値を読み出しもしくは書き込みたい場合はリストIII-32 のようにします。

ここで指定するレジスタの値は，00H から 0FH までです。また下位アドレスは使用(デ

リストIII-32

```
        LD      BC,1C00H  ┐
        LD      A,reg     │レジスタの指定
        OUT     (C),A     ┘
        LD      BC,1B00H  ┐読み出し
        IN      A,(C)     ┘
;
;       (EXIT)
;


        LD      BC,1C00H  ┐
        LD      A,reg     │レジスタの指定
        OUT     (C),A     ┘
        LD      BC,1B00H  ┐
        LD      A,data    │書き込み
        OUT     (C),A     ┘
;
;       (EXIT)
;
```

コード)していないので，Cレジスタはどんな値でもかまいません。

ふた組のI/Oポートには，単方向バッファが設けられていないため入力も出力も設定できます。したがって，この端子はジョイスティックの入力端子としてだけでなく，汎用のI/Oポートとしても利用できます。

図III-80　I/OポートとPSGの制御

システムI/Oポート	BDIR	BC1	機　　　能	I/O
——	0	0	PSGノンアクティブ	—
(IB**)H	0	1	PSGから読み出し	IN
	1	0	PSGへ書き込み	OUT
(IC**)H	1	1	レジスタの指定	OUT

*印は無効デジット

■レジスタの意味と設定法

16個のレジスタ(R_0～R_{15})の働きを詳しくみていきます。

トーンジェネレータ(A, B, C)

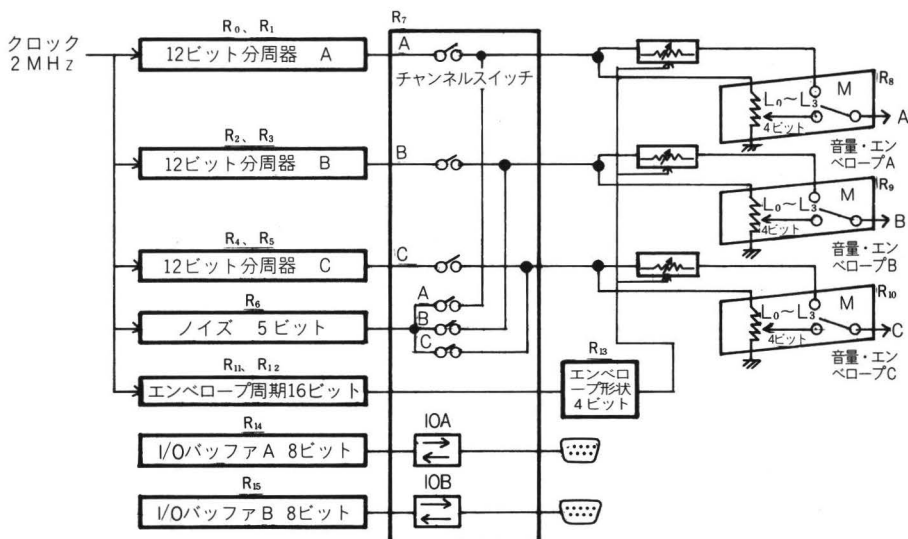
チャンネルA(R_0, R_1)

チャンネルB(R_2, R_3)

チャンネルC(R_4, R_5)

AY-3-8910内の各レジスタの関係を示したブロック図が図III-81です。サウンドIC内で作られるサウンドは，すべて供給クロックを分周したものを利用して作られます。X1ではこの供給クロックは，CPUクロックの1/2の2MHzです。

AY-3-8910に内蔵されている3つのトーンジェネレータ(A,B,C)は，12ビットの分周器

図III-81　AY-3-8910のレジスタ相関図

とみることができ，レジスタに設定する値が大きいほど低い音になります。このうち上位4ビットと下位8ビットをそれぞれR_1，R_0(チャンネルA)，R_3，R_2(チャンネルB)，R_5，R_4(チャンネルC)に設定することによって3種類のトーンを発生させることができます。

これをまとめると図III-82の表のようになります。

次のようにして分周比を計算し，トーンジェネレータから目的の周波数の音を出します。出そうと思う音の周波数を仮に440Hzとし，チャンネルAから出す場合を考えてみます。レジスタに設定する分周比TPは，

$f_t = 440\text{Hz}$　　f_t：出力する音の周波数

$f_{clock} = 2\text{MHz}$　　f_{clock}：供給クロックの周波数　(f_{clock}＝定数)

$= 2000000\text{Hz}$

とすると，

$$Tp = \frac{f_{clock}}{16 \times f_t} = \frac{2000000}{16 \times 440} \fallingdotseq 284$$

ここで分母に16をかけているのは，サウンドIC内部でさらに16分周されるからです。

得られたTP(分周比)からレジスタに設定する16進値を求めます。

284は2進数と16進数で表すとそれぞれ，

$000100011100_{(2)}$

011CH

となります。

10進数の2進数への変換法を参考のために少しだけ示しておきましょう。

```
2) 284 …… 0
2) 142 …… 0
2)  71 …… 1
2)  35 …… 1
2)  17 …… 1
2)   8 …… 0
2)   4 …… 0
2)   2 …… 0
2)   1 …… 1
     0
```

図III-82　トーン周波数構成(12ビット)

チャンネル	粗調整レジスタ	微調整レジスタ
A	R_1	R_0
B	R_3	R_2
C	R_5	R_4
ビット構成	C7 C6 C5 C4 C3 C2 C1 C0 (未使用)	F7 F6 F5 F4 F3 F2 F1 F0
	TP11 TP10 TP9 TP8 TP7 TP6 TP5 TP4 TP3 TP2 TP1 TP0 12ビット トーン周波数 最小値：000000000001(1)　最大値：111111111111($4{,}095_{10}$)	

このように“2”で割っていき，余りのところを下から並べればよいのです。12ビットということですから，上位に3つ0をつけ加えます。

これを4ビットと8ビットに分けR_1・R_0に設定します。チャンネルB，Cから音を出す場合も同様にR_3・R_2，R_5・R_4に設定します。

R_1＝01H

R_0＝1CH

実際にこの値を設定してみましょう。リストⅢ-33のプログラムを実行してください。

リストⅢ-33

```
                         ORG     0A000H

A000    01 00 1C         LD      BC,1C00H  ┐
A003    3E 00            LD      A,00H     │
A005    ED 79            OUT     (C),A     │
A007    01 00 1B         LD      BC,1B00H  │ R0 に 1CH を設定
A00A    3E 1C            LD      A,1CH     │
A00C    ED 79            OUT     (C),A     ┘
A00E    01 00 1C         LD      BC,1C00H  ┐
A011    3E 01            LD      A,01H     │
A013    ED 79            OUT     (C),A     │
A015    01 00 1B         LD      BC,1B00H  │ R1 に01H を設定
A018    3E 01            LD      A,01H     │
A01A    ED 79            OUT     (C),A     │
A01C    C9               RET               ┘
```

しかし，このままではサウンドICからは何も聞こえません。音を出すにはチャンネルAの音量レジスタR_8を設定した後，音の出力開始のためのチャンネルスイッチR_7を設定する必要があります。

ノイズジェネレータ

レジスタ(R_6)

このレジスタによってノイズの周波数が決まります。R_6は下位5ビットが有効で，このレジスタに設定する値を00H($00000_{(2)}$)から1FH($11111_{(2)}$)まで変化させることによって，シーという音からゴーという音まで出すことができます。

エンベロープと組み合わせると，時間的に変化する音，たとえば爆発音，ミサイルの発射音，波の音のような効果音も容易に作ることができよく利用されています。

ノイズは専用のボリュウムを持たないので，A，B，Cのチャンネルに乗せて出すことになります。

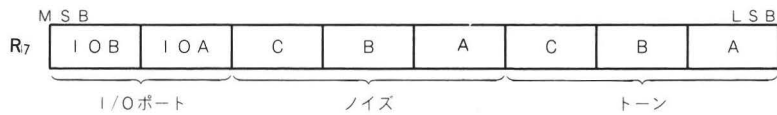
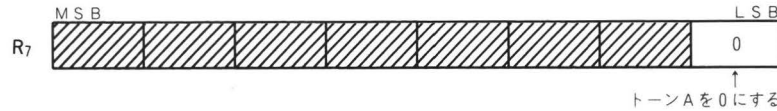
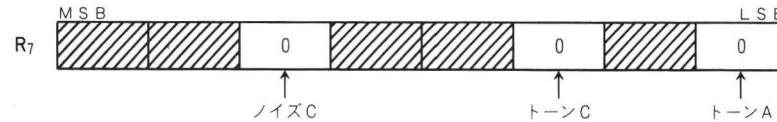
チャンネルスイッチ

レジスタ(R_7)

このレジスタはA，B，Cのトーンチャンネルの ON/OFF，ノイズをどのチャンネルに乗せるか，ジョイスティックのI/Oの方向などを決めます。

R_7の各ビットは，図Ⅲ-83のような意味を持っています。サウンドの場合，出力させたいチャンネルを0にします。

たとえば，チャンネルAからトーンを出したい場合は図III-84のようにし，チャンネルAからトーンを，チャンネルCからノイズとトーンを出したい場合は図III-85のようにR_7を設定します。

図III-83　R7ビット内容

図III-84　チャンネルAトーン出力例

図III-85　チャンネルA(トーン)・C(トーン，ノイズ)出力例

ただし，すでにある音を出している場合などは目的以外のビット(▨)が変化しないよう，R_7の値を読み出して，これをビットセット，リセット命令などによりビット設定をし，再びR_7に書き込みます。

Aチャンネルからトーンを出す場合の例をリストIII-34に示します。

リストIII-34

```
                        ORG     0A020H

A020    01 00 1C        LD      BC,1C00H
A023    3E 07           LD      A,07H
A025    ED 79           OUT     (C),A
A027    01 00 1B        LD      BC,1B00H
A02A    ED 78           IN      A,(C)     : R7の内容を読み出す
A02C    CB 87           RES     0,A       : Bit0をリセットする
A02E    ED 79           OUT     (C),A     : R7に書き込む
A030    C9              RET
```

リストIII-33に続いて実行してください。スピーカーから440Hzのトーンが聞こえるはずです。もし聞こえない場合は，さらに次に述べるチャンネルA音量レジスタ(R_8)を適当に設定してください。

一方ジョイスティック端子の入出力は，R_7のビット6，7で設定します。このビットは0にすると入力，1にすると出力になります。通常はジョイステック端子として用いるので，ともに0(入力)に設定してあります。

このふたつのポートのI/OバッファはR_{14}(ポートA)，R_{15}(ポートB)です。I/Oポート

の入出力を設定する場合にも，R_7の他のビットが変化しないようにしてください。

ポートA, Bを入出力に設定するサンプル例をリストIII-35 に示します。

ここでデータとしてセットする値は，図III-86 の表のようになります。

リストIII-35

```
                                ORG     0A000H

A000    01 00 1C                LD      BC,1C00H  ┐
A003    3E 07                   LD      A,07H     │レジスタ7を指定する
A005    ED 79                   OUT     (C),A     ┘
A007    01 00 1B                LD      BC,1B00H
A00A    ED 78                   IN      A,(C)     :レジスタの内容をAccに読み出す
A00C    E6 3F                   AND     3FH       :ビット6,7をリセット
A00E    F6 **                   OR      **        :入出力を設定するデータ**
A010    ED 79                   OUT     (C),A
A012    C9                      RET
```

図III-86　PSGポートA，Bの入出力モード設定データ(＊＊)

＊＊(16進)	ポートA ジョイスティック1	ポートB ジョイスティック2
0 0	IN	IN
4 0	OUT	IN
8 0	IN	OUT
C 0	OUT	OUT

音量レジスタとエンベロープスイッチ

チャンネルA(R_8)

チャンネルB(R_9)

チャンネルC(R_{10})

チャンネルA,B,Cの音量はそれぞれR_8, R_9, R_{10},の下位4ビット(bit0～3)により設定します。音量は16段階に設定でき，設定する値が大きいほど音量が大きくなります。音量を設定する場合は，ビット4は0にしておいてください。

チャンネルAを最大音量に設定する例をリストIII-36 に示します。

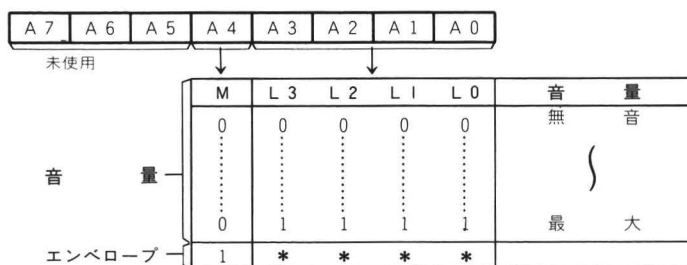
リストIII-36

```
                                ORG     0A040H

A040    01 00 1C                LD      BC,1C00H
A043    3E 08                   LD      A,08H
A045    ED 79                   OUT     (C),A
A047    01 00 1B                LD      BC,1B00H
A04A    3E 0F                   LD      A,0FH
A04C    ED 79                   OUT     (C),A
A04E    C9                      RET
```

ビット4はエンベロープ(音量の時間的変化)を用いるときに使うスイッチで，これを1にするとエンベロープモードになります。このとき音量レジスタに何が入っていても無視され，エンベロープが優先されます。

チャンネルB, Cも同様にしてR_9, R_{10}を設定します。図III-87 の表はその様子をまとめたものです。

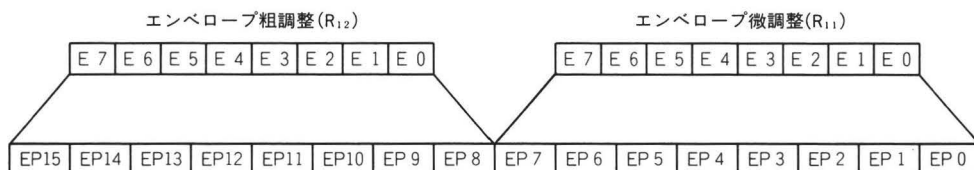
図III-87　音量とエンベロープ設定

＊印は無効（何が入っていても無視される）を意味する

エンベロープジェネレータ

エンベロープ周期(R_{11}, R_{12})

エンベロープ形状(R_{13})

エンベロープ出力の設定は，エンベロープジェネレータ R_{11}, R_{12}, R_{13}によって行われます。このうちエンベロープの繰り返しの周期を決めるのが R_{11}, R_{12}で，16ビット分周器のパラメータとなります。上位8ビットを R_{12}，下位8ビットを R_{11}で設定します(図III-88)。

図III-88　エンベロープ周期構成(16ビット)

一方，R_{13}はエンベロープの形状を設定します。

R_{11}，R_{12}に設定する分周比は次のようにして計算します。

$$E_P=\frac{f_{clock}}{256\times f_E}$$

f_{clock}：供給クロック(2MHz)
f_E　：設定する周波数
E_P　：エンベロープ周期

たとえば，繰り返しの周期5秒，つまり周波数を1/5Hzとしますと，Epは次の値になります。

$$E_P=\frac{2\times10^6}{256\times1/5}\fallingdotseq39063$$

ここで分母に256をかけているのは，サウンドIC内でさらに256分周(2^8)されるからです。トーンジェネレータのときはこれは16でした。

求めた分周比は16進数に変換して R_{12}と R_{13}に設定します。こんどは手作業でなくHu-BASICのHEX$関数(たまには思い出しましょう)を使ってみましょう。

Hu-BASICを起動し，

```
?HEX$(39063)
9897
OK
```

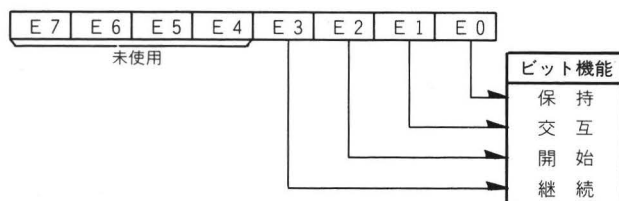
とするとレジスタに格納する16進数が簡単に求められます。

R_{12}＝98H

R_{11}＝97H

を設定し，各チャンネルのエンベロープスイッチ(R_8，R_9，R_{10}のビット4)を1に設定すると，約5秒の周期で音量が変化するサウンドが得られます。

エンベロープの形状の設定は，R_{13}によって行いますが，R_{13}の下位4ビットが有効で，各ビットはそれぞれ図III-89のような意味を持っています(実用上は，その次の波形だけを考えるので十分です)。

図III-89　エンベロープ形状設定・ビット構成

保持ビットと交互ビットの両者が論理"1"のとき，エンベロープカウンタは保持状態に入る前に，初期カウンタ値へリセットされる。

保持(Hold)：1のとき，カウントアップ/ダウンモードによって，そのエンベロープカウンタの最終値(E3～E0＝0000または1111)を保持します。

交互(Alternate)：1のとき，最初のサイクルがカウントダウンであればその次のサイク

図III-90　各種エンベロープ波形

E_3 CONTINUE	E_2 ATTACK	E_1 ALTERNATE	E_0 HOLD	出力波形
0	0	X	X	
0	1	X	X	
1	0	0	0	
1	0	0	1	
1	0	1	0	右図(拡大図)
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

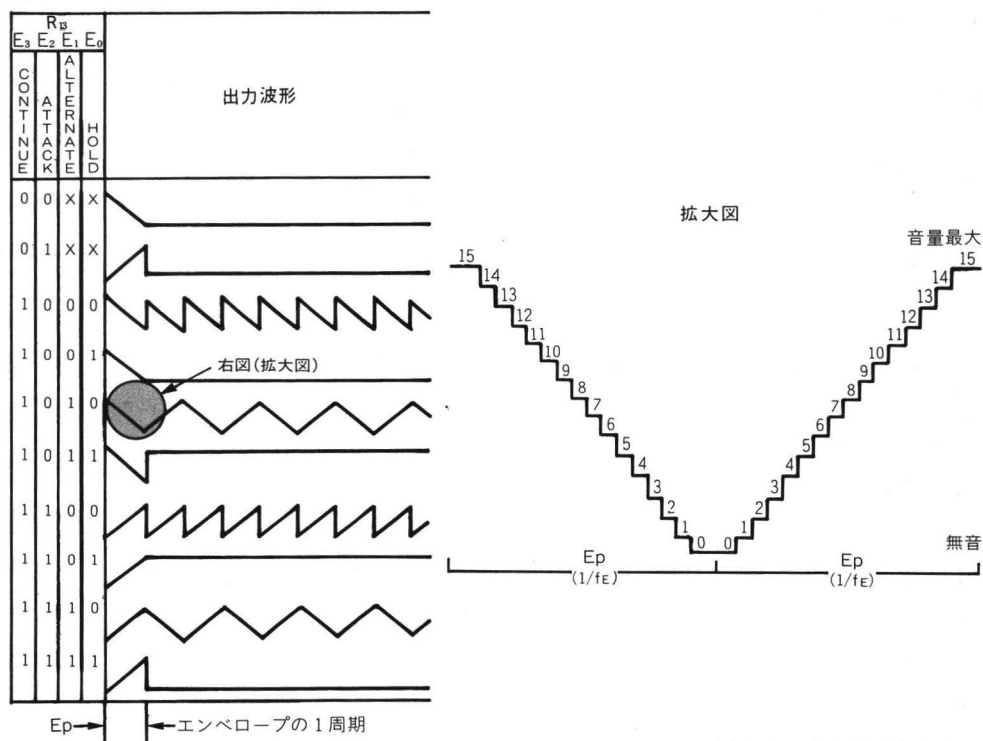
Ep→　←エンベロープの1周期

拡大図

音量最大

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

無音

Ep (1/fE)　Ep (1/fE)

ルは，カウントアップというような操作を交互に実行します。

開始(Attack)：1のとき，エンベロープカウンタは，0000から1111までカウントアップを実行，0のとき，1111から0000までカウントダウンを実行します。

継続(Continue)：1のとき，サイクルパターンは，保持ビットの内容に依存。0のとき，1サイクル後，0000値へリセットされ，この値を保持します。

これらの各ビットの組み合わせによって，図III-90のような波形が得られます。

次に，実際にいろいろなエンベロープを使って，トーンに変調をかけてみましょう。

例としてAチャンネルのトーンにエンベロープを使うことにします。リストIII-33,34に続いてリストIII-37を実行してみてください。なお，これはBASICのSOUNDコマンドを使えば(より簡単に)実行できますので試してみてください。

リストIII-37

```
                        ORG     0A040H       **=出力データ

A040    01 00 1C        LD      BC,1C00H  ┐
A043    3E 08           LD      A,08H     │ ①
A045    ED 79           OUT     (C),A     │ チャンネルA のエンベロープスイッチを1にする
A047    01 00 1B        LD      BC,1B00H  │
A04A    3E 10           LD      A,10H     │
A04C    ED 79           OUT     (C),A     ┘
A04E    01 00 1C        LD      BC,1C00H  ┐
A051    3E 0B           LD      A,0BH     │
A053    ED 79           OUT     (C),A     │
A055    01 00 1B        LD      BC,1B00H  │
A058    3E 93           LD      A,93H     │ ②
A05A    ED 79           OUT     (C),A     │ 繰り返しの周期を約5秒とする
A05C    01 00 1C        LD      BC,1C00H  │
A05F    3E 0C           LD      A,0CH     │
A061    ED 79           OUT     (C),A     │
A063    01 00 1B        LD      BC,1B00H  │
A066    3E 98           L       A,98H     │
A068    ED 79           OUT     (C),A     ┘
A06A    01 00 1C        LD      BC,1C00H  ┐
A06D    3E 0D           LD      A,0DH     │ ③
A06F    ED 79           OUT     (C),A     │ エンベロープ形状を設定する
A071    01 00 1B        LD      BC,1B00H  │
A074    3E **           LD      A,**      │
A076    ED 79           OUT     (C),A     │
A078    C9              RET               ┘
```

ジョイスティックポート

I/Oポート$A(R_{14})$

I/Oポート$B(R_{15})$

ジョイスティック端子としての使い方

ポートA，Bをジョイスティック用入力端子として使うには，これを入力モードに設定する必要があります。ポートA，Bの入出力の設定はリストIII-35を参考にしてください。

A，Bともに入力に設定する場合，R_7のビット6，7をともに0に設定します。データはジョイスティック1がR_{14}，ジョイスティック2がR_{15}にそれぞれ記憶されているので，そのデータはR_{14},R_{15}を読むことで得られます。

R_{14},R_{15}は，サウンドとは無関係に設定と読み出しができるので，このレジスタを操作することによって音が変化することはありません。

例としてジョイスティック1のデータを読むプログラムをリストIII-38に示します。

```
リストIII-38
                                ORG     0A000H

A000    01 00 1C                LD      BC,1C00H
A003    3E 0E                   LD      A,0EH
A005    ED 79                   OUT     (C),A
A007    01 00 1B                LD      BC,1B00H
A00A    ED 78                   IN      A,(C)
A00C    C9                      RET
```

このルーチンを実行すると，ジョイスティックのデータがAccに取り込まれます。データの各ビットは図III-91のような意味を持っています。

図III-91　ジョイスティックデータ・ビット構成

ビット7 MSB	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0 LSB
トリガボタン2(機種による)	トリガボタン2(MSX)	トリガボタン1	未使用	⇨	⇦	⇩	⇧

ジョイスティックが何も押されていないと各ビットがすべて1になり，押されると対応するビットが0になります。また，斜めの判断(たとえば⇗)の場合，ビット0とビット3が0になります。

ジョイスティックデータの処理方法で注意しなければならないことがあります。それは，ゲームなどでバーの移動とトリガボタン入力を同時に行った場合には，上位と下位の4ビットが独立した値をとるので判断がやや面倒になるということです。したがって単純に比較命令(CP n)でデータを処理せずにビットテストを用いるか，AND命令で上位4ビットあるいは下位4ビットをマスクしてから処理するなどの工夫が必要です。

リストIII-39, 40に移動方向データを読む場合およびトリガボタンを読む場合のジョイスティックデータの読み込み処理ルーチンの例を示します。

```
リストIII-39
        移動方向データを読む場合
        LD      BC,1C00H  ┐
        LD      A,0EH     │ジョイスティック1のデータを読む
        OUT     (C),A     │
        LD      BC,1B00H  │
        IN      A,(C)     ┘
        AND     0FH       ;上位4ビットをマスクする
        CP      data      ;比較データ
        JR      Z,MOVEn   ;処理ルーチンへジャンプ
  :
  :
  :     (EXIT)
```

汎用I/Oポートとしての使い方

ジョイスティック端子1，2は，ともに8ビットの汎用I/Oポートとしても利用できます。ポートの入出力の設定は，リストIII-35を参考にしてください。

ポートを出力として利用する場合R_{14},R_{15}はバッファレジスタとして働くので，一度データを設定すると，次のデータがセットされるまでその値を保持します。

I/OポートAを出力，Bを入力に設定し，8×8マトリクス(64キー)の外部キーボードを読み取るルーチンをリストIII-41に示します。

リストⅢ-40

トリガボタンを読む場合

```
        LD     BC,1C00H  ┐
        LD     A,0EH     │
        OUT    (C),A     │ジョイスティック1のデータを読む
        LD     BC,1B00H  │
        IN     A,(C)     ┘
        OR     0FH       :下位4ビットをマスクする
        INC    A         :もしFFH（何も押されていない）ならインクリメントすると00H
        JR     NZ,STRIG  :トリガ入力の処理
;
;       (EXIT)
;
```

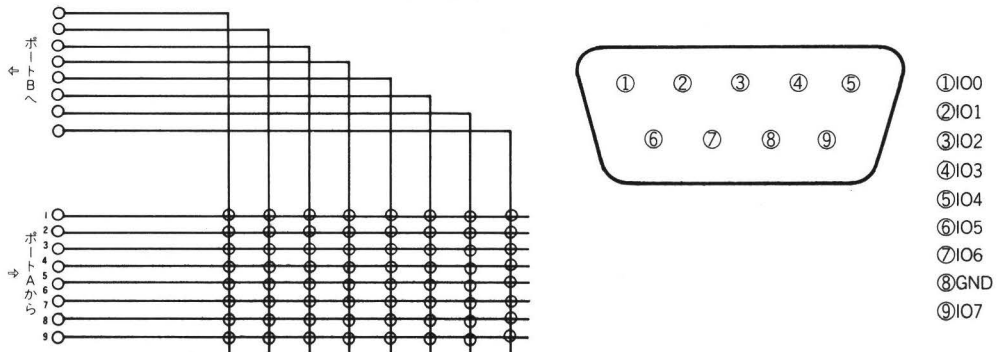
リストⅢ-41

8×8マトリクス読み取りルーチン

```
                                ORG    0A000H

A000    01 00 1C                LD     BC,1C00H  ┐
A003    3E 07                   LD     A,07H     │
A005    ED 79                   OUT    (C),A     │
A007    01 00 1B                LD     BC,1B00H  │
A00A    ED 78                   IN     A,(C)     │ ①
A00C    E6 3F                   AND    3FH       │
A00E    F6 40                   OR     40H       │
A010    ED 79                   OUT    (C),A     │
A012    37                      SCF              ┘
A013    16 FE                   LD     D,0FEH    ┐
A015    01 00 1C        SRCHLP: LD     BC,1C00H  │
A018    3E 0E                   LD     A,0EH     │ ②
A01A    ED 79                   OUT    (C),A     │
A01C    01 00 1B                LD     BC,1B00H  │
A01F    ED 51                   OUT    (C),D     ┘
A021    01 00 1C                LD     BC,1C00H  ┐
A024    3E 0F                   LD     A,0FH     │
A026    ED 79                   OUT    (C),A     │ ③
A028    01 00 1B                LD     BC,1B00H  │
A02B    ED 78                   IN     A,(C)     ┘
A02D    5F                      LD     E,A
A02E    3C                      INC    A
A02F    20 04                   JR     NZ,EXIT   :④
A031    CB 12                   RL     D         :⑤
A033    38 E0                   JR     C,SRCHLP  :⑥
A035    C9              EXIT:   RET
```

① ポートA（ジョイスティック1）出力，ポートB（ジョイスティック2）入力に設定
② サーチするラインを0にする，それ以外は1
③ ポートBからキーデータを読む
④ キーが何か押されておればリターン
⑤ サーチするラインを移す
⑥ 8本ともサーチし終わればリターン

図Ⅲ-92 リストⅢ-41使用キーボード回路図

このルーチンを実行すると，サーチしたラインがDレジスタに，キーデータがEレジスタに入ります。

データは，キーが押しているラインおよびデータのビットが0になります。この例で用いた外部キーボードの回路図は図III-92のとおりです。

効果的なサウンド作り

■音階データの作成法

PSGは高機能のICなので，最大限利用して面白い効果を引き出したいものです。

PSGの使い方としていくつかの応用例をあげてみましょう。

PSGは，3重和音を活かした音楽演奏が楽しめます。音階の基本に少しふれたあと音階データの作成法をみてみます。

1オクターブの差は，周波数にしてちょうど2倍にあたります。たとえば低い“ラ”を440Hzとすると，高い“ラ”は880Hzです。この440Hzというのは世界的に決められている“ラ”の周波数です。

1オクターブは12の音階に分けられ，この音階を平均に割りふったものを平均律音階といいます。音階と音階の比が12音ともすべて等しいので，これを12回かけ合わせるとちょうど2になるように決められているのです。

つまり，1/12オクターブ(半音階)は，

$$eXp(\ln 2/12) = 1.059462$$

になるわけです。したがって分周比は半音ごとに1.059462倍になるように設定すればよいことになります。

ドレミファソラシドを平均律で計算したのが図III-93のⓐⓑです。ところがこの表で計算した分周比で3重和音を出すと，非常ににごった音になります。

これは音階比は本来簡単な整数比にならなければいけないのですが，あちらを立てればこちらが立たずでうまくいかないので，便宜上平均律を使っているからなのです。さらにPSGの出力は矩形波のため強い3倍音を含んでいるので，分周比が整数比でないと汚いビートを発生するのです。

そこで音階を純正律で計算すると，意外にきれいで力強い和音が出ますので試してみてください。図III-93のⓒⓓが音階比と分周比です。ただし純正律は移調すると音階がくずれます。いろいろと微妙な問題もあるので和音を作る場合は，最適な分周比をそのつど計算してください。

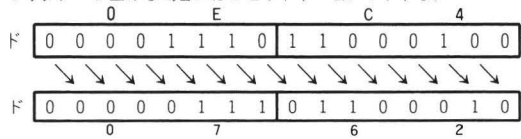
分周比は，各音階の一番低い音を出す場合のデータですから，これより1オクターブ高い音を出したい場合は，1ビットずつ右シフトさせてください。このことは2進法では「2で割る」ことで，分周比が半分になるので周波数が2倍になります。

LSBが1の場合，切り上げるか切り捨てるかは考えるところです。きれいな和音を作るには，和音の分周比が1.20倍か，1.25倍に近くなるように選ぶとよいでしょう。

図III-93　平均律と純正律

音階		ⓐ 平均律 音階比	ⓑ 平均律 分周比(HEX)	ⓒ 純正律 音階比	ⓓ 純正律 分周比(HEX)
C	ド	1.000000	0EEF	1.000000	0EC4
D	レ	1.122460	0D4D	1.125000	0D20
E	ミ	1.259916	0BF2	1.250000	0BD0
F	ファ	1.334841	0B2F	1.350000	0AF0
G	ソ	1.498305	09F5	1.500000	09D8
A	ラ	1.681777	08E1	1.687500	08C0
B	シ	1.887727	07E9	1.875000	07E0
C	ド	2.000000	0778	2.000000	0762
			(D_2)	2.250000	

1オクターブ上げる場合には1ビットずつ右シフトする。

```
       0        E        C        4
ド  0 0 0 0 1 1 1 0 | 1 1 0 0 0 1 0 0
    ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘
ド  0 0 0 0 0 1 1 1 | 0 1 1 0 0 0 1 0
       0        7        6        2
```

サンプルとして電子(キーボード)オルガンのプログラムをリストIII-42に紹介します。キーボードをオルガンの鍵盤のように使って演奏できるようにしたものです。キーの同時入力をIOCS内の割り込みによるキー処理によって検出しているので，その例としても参考になるでしょう。

リストIII-42

```
                                ORG     0A000H

A000    AF                      XOR     A
A001    32 26 A0                LD      (CHNPTR),A
A004    21 38 A0                LD      HL,KEYACS
A007    22 52 00                LD      (0052H),HL
A00A    21 3F 07                LD      HL,073FH
A00D    CD 28 A0                CALL    PSGSET
A010    21 0F 08                LD      HL,080FH
A013    CD 28 A0                CALL    PSGSET
A016    24                      INC     H
A017    CD 28 A0                CALL    PSGSET
A01A    24                      INC     H
A01B    CD 28 A0                CALL    PSGSET
A01E    06 00           ELOOP:  LD      B,00H
A020    10 FC                   DJNZ    ELOOP
A022    18 FA                   JR      ELOOP
A024    00                      NOP
A025    00                      NOP
A026    00              CHNPTR: NOP
A027    00                      NOP
A028    06 1C           PSGSET: LD      B,1CH
A02A    ED 61                   OUT     (C),H
A02C    05                      DEC     B
A02D    ED 69                   OUT     (C),L
A02F    C9                      RET
A030    06 1C           PSGRD:  LD      B,1CH
A032    ED 61                   OUT     (C),H
A034    05                      DEC     B
A035    ED 78                   IN      A,(C)
A037    C9                      RET
A038    F3              KEYACS: DI
A039    CD 49 0B                CALL    0B49H
A03C    E6 20                   AND     20H
A03E    20 05                   JR      NZ,KDATRD
A040    CD 49 0B                CALL    0B49H
A043    FB                      EI
A044    C9                      RET
A045    CD 49 0B        KDATRD: CALL    0B49H
A048    FE 41                   CP      41H
A04A    11 B1 03                LD      DE,03B1H
A04D    28 42                   JR      Z,PTRNXT
A04F    FE 53                   CP      53H
A051    11 48 03                LD      DE,0348H
A054    28 3B                   JR      Z,PTRNXT
A056    FE 44                   CP      44H
A058    11 F4 02                LD      DE,02F4H
```

```
A05B    28 34                JR     Z,PTRNXT
A05D    FE 46                CP     46H
A05F    11 BC 02             LD     DE,02BCH
A062    28 2D                JR     Z,PTRNXT
A064    FE 47                CP     47H
A066    11 76 02             LD     0276H
A069    28 26                JR     Z,PTRNXT
A06B    FE 48                CP     48H
A06D    11 30 02             LD     DE,0230H
A070    28 1F                JR     Z,PTRNXT
A072    FE 4A                CP     4AH
A074    11 F8 01             LD     DE,01F8H
A077    28 18                JR     Z,PTRNXT
A079    FE 4B                CP     4BH
A07B    11 D9 01             LD     DE,01D9H
A07E    28 11                JR     Z,PTRNXT
A080    FE 00                CP     00H
A082    20 06                JR     NZ,SHFTBK
A084    21 3F 07             LD     HL,073FH
A087    CD 28 A0             CALL   PSGSET
A08A    FE 03       SHFTBK:  CP     03H
A08C    CA 00 00             JP     0000H
A08F    FB                   EI
A090    C9                   RET
A091    3A 26 A0    PTRNXT:  LD     A,(CHNPTR)
A094    FE 00                CP     00H
A096    26 00                LD     H,00H
A098    28 08                JR     Z,PTRLB
A09A    FE 01                CP     01H
A09C    26 02                LD     H,02H
A09E    28 02                JR     Z,PTRLB
A0A0    26 04                LD     H,04H
A0A2    3C          PTRLB:   INC    A
A0A3    FE 03                CP     03H
A0A5    20 01                JR     NZ,CHNSET
A0A7    AF                   XOR    A
A0A8    32 26 A0    CHNSET:  LD     (A026H),A
A0AB    6B                   LD     L,E
A0AC    CD 28 A0             CALL   PSGSET
A0AF    24                   INC    H
A0B0    6A                   LD     L,D
A0B1    CD 28 A0             CALL   PSGSET
A0B4    26 07                LD     H,07H
A0B6    3A 26 A0             LD     A,(CHNPTR)
A0B9    6F                   LD     L,A
A0BA    B7                   OR     A
A0BB    20 02                JR     NZ,CHNOPN
A0BD    2E 04                LD     L,04H
A0BF    CD 30 A0    CHNOPN:  CALL   PSGRD
A0C2    B5                   OR     L
A0C3    57                   LD     D,A
A0C4    7D                   LD     A,L
A0C5    2F                   CPL
A0C6    A2                   AND    D
A0C7    6F                   LD     L,A
A0C8    CD 28 A0             CALL   PSGSET
A0CB    FB                   EI
A0CC    C9                   RET
```

和音は最大3重和音まででです。キー入力があるたびに，設定するチャンネルをA→B→C→Aと変えてゆくと，たくさんのサウンドチャンネルがあるように聞こえます。

音階データは8個のキー(キーボード上AからKまでの横1列)に割り付けられています。しかし，たった1オクターブではさびしいので，興味のある方はあなたの手で完全なものに仕上げてください。音階データは図III-93のものより2オクターブ上げてあります。この部分にいろいろな分周比を設定してよい和音を作ってください。

■効果音の作り方

実際にサウンドICの各機能を活かした効果音の具体例をいくつか見ていくことにしましょう。

サウンドICは3チャンネルのトーンのほかに，1チャンネルのノイズと1チャンネルのエンベロープが使えます。これらの使い方の例として，波の音のプログラムをリストIII-43に示します。

Aチャンネルからノイズを出し，これにエンベロープモードで変調をかけます。波らしい感じを出すために，Bチャンネルからも小音量でノイズを出しています。

リストIII-43

```
                                      ORG       0A000H
A000    21 18 A0                      LD        HL,REGDAT
A003    16 00                         LD        D,00H
A005    06 1C             SETLOP:     LD        B,1CH
A007    7A                            LD        A,D
A008    ED 79                         OUT       (C),A
A00A    05                            DEC       B
A00B    7E                            LD        A,(HL)
A00C    ED 79                         OUT       (C),A
A00E    23                            INC       HL
A00F    14                            INC       D
A010    7A                            LD        A,D
A011    FE 10                         CP        10H
A013    20 F0                         JR        NZ,SETLOP
A015    C9                            RET
A016    00                            NOP
A017    00                            NOP
                                      ;
A018    00                REGDAT:     DB        00H
A019    00                            DB        00H
A01A    00                            DB        00H
A01B    00                            DB        00H
A01C    00                            DB        00H
A01D    00                            DB        00H
A01E    1F                            DB        1FH         :①
A01F    27                            DB        27H         :②
A020    10                            DB        10H         :③
A021    06                            DB        06H         :④
A022    00                            DB        00H
A023    00                            DB        00H
A024    F0                            DB        0F0H        :⑤
A025    0A                            DB        0AH         :⑥
A026    FF                            DB        0FFH
A027    FF                            DB        0FFH
```

① ノイズの音色　　④ Bチャンネル音量
② ノイズをA，Bチャンネルに乗せる　　⑤ 波の周期
③ Aチャンネルエンベロープモード　　⑥ エンベロープ形状

この例では，サウンドIC内の16個のレジスタすべてに値を設定しています。しかし必ずしもそのようにしなければならないというわけではありません。サウンドIC内の内部構造をよく考えたうえで，必要なレジスタだけに値を設定すればよいでしょう。

リストIII-43ではA018HからA027Hまでが，レジスタに設定するデータのテーブルになっています。この部分をいろいろと変えてみましょう。

リストIII-44のⓐはピストルの音です。エンベロープモードとして09Hを設定すると爆発音の感じが出ます。レジスタの内容をその後変更していない場合，ふたたび同じ爆発音を出すときはすべてのレジスタを設定し直してもよいのですが，レジスタ13を設定するだけでも出すことができます。

リストIII-44のⓑは大砲の音です。ⓐの音程をもっとも低く設定すると重い爆発音になります。チャンネルAからさらにトーンを重ねると，より面白い効果音になります(リストIII-44 ⓒ)。

ノイズを早い周期で変調すると，戦車のキャタピラ音になります(リストIII-44 ⓓ)。次にジェット音を作ってみたのがリストIII-44 ⓔです。トーンとノイズの混合だとトーンの音が純粋すぎ，ジェット音らしくありません。ノイズを変調させることによって，ジェット音らしい音が得られます。

このほかにも，いろいろな効果音が作れますから，レジスタに設定する値を変えて試してみてください。

リストⅢ-44　　効果音データ

```
:A018=00 00 00 00 00 00 08 37  :  ピストルの音  ⓐ
:A020=10 00 00 00 40 09 FF FF

:A018=00 00 00 00 00 00 1F 37  :  大砲の音  ⓑ
:A020=10 00 00 00 40 09 FF FF

:A018=80 00 00 00 00 00 08 36  :  ミサイルの発射音  ⓒ
:A020=10 00 00 00 40 09 FF FF

:A018=00 00 00 00 00 00 08 37  :  キャタピラー音  ⓓ
:A020=10 00 00 00 01 0C FF FF

:A018=00 00 00 00 00 00 01 37  :  ジェットの音  ⓔ
:A020=10 00 00 01 00 0A FF FF
```

■残響効果を出す例

こんどはノイズとエンベロープを組み合わせて，残響効果を出してみます。チャンネルから出ている音を急に切らずに，エンベロープを使ってしだいに音を小さくしていくと，いかにも残響のような効果が出るというわけです。

残響効果をうまく使うと，トーンにも丸みが出せますし広い部屋で鳴らしているような感じにもなります。

リストⅢ-45，46は，この残響効果を使った電子オルガンのプログラム例です。この例でも，割り込みによるキー入力処理を用いて入力を検出しています。

キーボードから割り込みがかかるごとに，トーンを出すチャンネルをA→B→C→Aと移していきます。新しいキー入力があるか，キーボードから00Hが送られてくると，それまで鳴っていたチャンネルを閉じます。

このとき急に音を切らずに，エンベロープによってしだいに音を小さくするか，小音量に設定し直すなどして，それまで鳴っていたチャンネルの音をわずかでも残すようにします。

リストⅢ-45は，エンベロープによって残響効果を出した例です。この方法は残響が強すぎる気もしますが，カノンのように和音を追いかける(和声的)メロディでは，すばらしい効果が期待できます。

なお音階をたどるような(旋律的)メロディのときは，隣りあう音が混って予測できない効果を生じ聞き苦しいので，小音量を残す方がよい効果が得られます。リストⅢ-46はこの方法を用いて残響効果を出した例です。残す音量は，メロディを最大音量(0FH)に設定するとして，0AHぐらいがよいでしょう。

リストIII-45

キーボードオルガンI

```
                                        ORG     0A000H
A000    AF                              XOR     A
A001    32 26 A0                        LD      (CHNPTR),A
A004    21 38 A0                        LD      HL,KEYACS
A007    22 52 00                        LD      (0052H),HL
A00A    21 60 0C                        LD      HL,0C60H
A00D    CD 28 A0                        CALL    PSGSET
A010    21 38 07                        LD      HL,0738H
A013    CD 28 A0                        CALL    PSGSET
A016    06 00               ELOOP:      LD      B,00H
A018    10 FC                           DJNZ    ELOOP
A01A    18 FA                           JR      ELOOP
A01C    00                              DB      00H
A01D    00                              DB      00H
A01E    00                              DB      00H
A01F    00                              DB      00H
A020    00                              DB      00H
A021    00                              DB      00H
A022    00                              DB      00H
A023    00                              DB      00H
A024    00                              DB      00H
A025    00                              DB      00H
A026    00                  CHNPTR:     NOP
A027    00                              NOP
A028    06 1C               PSGSET:     LD      B,1CH
A02A    ED 61                           OUT     (C),H
A02C    05                              DEC     B
A02D    ED 69                           OUT     (C),L
A02F    C9                              RET
A030    06 1C                           LD      B,1CH
A032    ED 61                           OUT     (C),H
A034    05                              DEC     B
A035    ED 78                           IN      A,(C)
A037    C9                              RET
A038    F3                  KEYACS:     DI
A039    CD 49 0B                        CALL    0B49H
A03C    E6 20                           AND     20H
A03E    20 05                           JR      NZ,KDATRD
A040    CD 49 0B                        CALL    0B49H
A043    FB                              EI
A044    C9                              RET
A045    CD 49 0B            KDATRD:     CALL    0B49H
A048    FE 41                           CP      41H
A04A    11 B1 03                        LD      DE,03B1H
A04D    28 42                           JR      Z,PTRNXT
A04F    FE 53                           CP      53H
A051    11 48 03                        LD      DE,0348H
A054    28 3B                           JR      Z,PTRNXT
A056    FE 44                           CP      44H
A058    11 F4 02                        LD      DE,02F4H
A05B    28 34                           JR      Z,PTRNXT
A05D    FE 46                           CP      46H
A05F    11 BC 02                        LD      DE,02BCH
A062    28 2D                           JR      Z,PTRNXT
A064    FE 47                           CP      47H
A066    11 76 02                        LD      DE,0276H
A069    28 26                           JR      Z,PTRNXT
A06B    FE 48                           CP      48H
A06D    11 30 02                        LD      DE,0230H
A070    28 1F                           JR      Z,PTRNXT
A072    FE 4A                           CP      4AH
A074    11 F8 01                        LD      DE,01F8H
A077    28 18                           JR      Z,PTRNXT
A079    FE 4B                           CP      4BH
A07B    11 D9 01                        LD      DE,01D9H
A07E    28 11                           JR      Z,PTRNXT
A080    FE 00                           CP      00H
A082    28 07                           JR      Z,RDEXT
A084    FE 03                           CP      03H
A086    CA 00 00                        JP      Z,0000H
A089    FB                              EI
A08A    C9                              RET
A08B    AF                  RDEXT:      XOR     A
A08C    32 27 A0                        LD      (CHNPTR+1),A
```

```
A08F    18 6F              JR     PSGL1
A091    3A 26 A0  PTRNXT:  LD     A,(CHNPTR)
A094    FE 00              CP     00H
A096    26 00              LD     H,00H
A098    28 08              JR     Z,PTRLB
A09A    FE 01              CP     01H
A09C    26 02              LD     H,02H
A09E    28 02              JR     Z,PTRLB
A0A0    26 04              LD     H,04H
A0A2    3C        PTRLB:   INC    A
A0A3    FE 03              CP     03H
A0A5    20 01              JR     NZ,PSGL2
A0A7    AF                 XOR    A
A0A8    32 26 A0  PSGL2:   LD     A,(CHNPTR)
A0AB    6B                 LD     L,E
A0AC    CD 28 A0           CALL   PSGSET
A0AF    24                 INC    H
A0B0    6A                 LD     L,D
A0B1    CD 28 A0           CALL   PSGSET
A0B4    3A 27 A0           LD     A,(CHNPTR+1)
A0B7    B7                 OR     A
A0B8    CA 50 A1           JP     Z,PSGL3
A0BB    3A 26 A0           LD     A,(CHNPTR)
A0BE    00                 NOP
A0BF    00                 NOP
A0C0    21 0F 08           LD     HL,080FH
A0C3    CD 28 A0           CALL   PSGSET
A0C6    21 00 09           LD     HL,0900H
A0C9    CD 28 A0           CALL   PSGSET
A0CC    21 10 0A           LD     HL,0A10H
A0CF    CD 28 A0           CALL   PSGSET
A0D2    FE 01              CP     01H
A0D4    28 28              JR     Z,PSGL4
A0D6    21 10 08           LD     HL,0810H
A0D9    CD 28 A0           CALL   PSGSET
A0DC    21 0F 09           LD     HL,090FH
A0DF    CD 28 A0           CALL   PSGSET
A0E2    21 00 0A           LD     HL,0A00H
A0E5    CD 28 A0           CALL   PSGSET
A0E8    FE 02              CP     02H
A0EA    28 12              JR     Z,PSGL4
A0EC    21 00 08           LD     HL,0800H
A0EF    CD 28 A0           CALL   PSGSET
A0F2    21 10 09           LD     HL,0910H
A0F5    CD 28 A0           CALL   PSGSET
A0F8    21 0F 0A           LD     HL,0A0FH
A0FB    CD 28 A0           CALL   PSGSET
A0FE    18 41     PSGL4:   JR     PSGL5
A100    3A 26 A0  PSGL1:   LD     A,(CHNPTR)
A103    21 10 08           LD     HL,0810H
A106    CD 28 A0           CALL   PSGSET
A109    21 00 09           LD     HL,0900H
A10C    CD 28 A0           CALL   PSGSET
A10F    21 00 0A           LD     HL,0A00H
A112    CD 28 A0           CALL   PSGSET
A115    FE 01              CP     01H
A117    28 28              JR     Z,PSGL5
A119    21 00 08           LD     HL,0800H
A11C    CD 28 A0           CALL   PSGSET
A11F    21 10 09           LD     HL,0910H
A122    CD 28 A0           CALL   PSGSET
A125    21 00 0A           LD     HL,0A00H
A128    CD 28 A0           CALL   PSGSET
A12B    FE 02              CP     02H
A12D    28 12              JR     Z,PSGL5
A12F    21 00 08           LD     HL,0800H
A132    CD 28 A0           CALL   PSGSET
A135    21 00 09           LD     HL,0900H
A138    CD 28 A0           CALL   PSGSET
A13B    21 10 0A           LD     HL,0A10H
A13E    CD 28 A0           CALL   PSGSET
A141    21 09 0D  PSGL5:   LD     HL,0D09H
A144    CD 28 A0           CALL   PSGSET
A147    FB                 EI
A148    C9                 RET
A149    00                 NOP
A14A    00                 NOP
A14B    00                 NOP
```

```
A14C	00		NOP
A14D	00		NOP
A14E	00		NOP
A14F	00		NOP
A150	3E 01	PSGL3:	LD	A,01H
A152	32 27 A0		LD	(CHNPTR+1),A
A155	3A 26 A0		LD	A,(CHNPTR)
A158	21 0F 08		LD	HL,080FH
A15B	CD 28 A0		CALL	PSGSET
A15E	21 00 09		LD	HL,0900H
A161	CD 28 A0		CALL	PSGSET
A164	21 10 0A		LD	HL,0A10H
A167	CD 28 A0		CALL	PSGSET
A16A	FE 01		CP	01H
A16C	28 28		JR	Z,PSGL6
A16E	21 10 08		LD	HL,0810H
A171	CD 28 A0		CALL	PSGSET
A174	21 0F 09		LD	HL,090FH
A177	CD 28 A0		CALL	PSGSET
A17A	21 00 0A		LD	HL,0A00H
A17D	CD 28 A0		CALL	PSGSET
A180	FE 02		CP	02H
A182	28 12		JR	Z,PSGL6
A184	21 00 08		LD	HL,0800H
A187	CD 28 A0		CALL	PSGSET
A18A	21 10 09		LD	HL,0910H
A18D	CD 28 A0		CALL	PSGSET
A190	21 0F 0A		LD	HL,0A0FH
A193	CD 28 A0		CALL	PSGSET
A196	FB	PSGL6:	EI
A197	C9		RET
```

リストⅢ-46

キーボードオルガン 2

```
			ORG	0A000H
A000	AF		XOR	A
A001	32 26 A0		LD	(CHNPTR),A
A004	21 38 A0		LD	HL,KEYACS
A007	22 52 00		LD	(0052H),HL
A00A	21 60 0C		LD	HL,0C60H
A00D	CD 28 A0		CALL	PSGSET
A010	21 38 07		LD	HL,0738H
A013	CD 28 A0		CALL	PSGSET
A016	06 00	ELOOP:	LD	B,00H
A018	10 FC		DJNZ	ELOOP
A01A	18 FA		JR	ELOOP
A01C	00		DB	00H
A01D	00		DB	00H
A01E	00		DB	00H
A01F	00		DB	00H
A020	00		DB	00H
A021	00		DB	00H
A022	00		DB	00H
A023	00		DB	00H
A024	00		DB	00H
A025	00		DB	00H
A026	00	CHNPTR:	NOP
A027	00		NOP
A028	06 1C	PSGSET:	LD	B,1CH
A02A	ED 61		OUT	(C),H
A02C	05		DEC	B
A02D	ED 69		OUT	(C),L
A02F	C9		RET
A030	06 1C		LD	B,1CH
A032	ED 61		OUT	(C),H
A034	05		DEC	B
A035	ED 78		IN	A,(C)
A037	C9		RET
A038	F3	KEYACS:	DI
A039	CD 49 0B		CALL	0B49H
```

```
A03C    E6 20                AND   20H
A03E    20 05                JR    NZ,KDATRD
A040    CD 49 0B             CALL  0B49H
A043    FB                   EI
A044    C9                   RET
A045    CD 49 0B    KDATRD:  CALL  0B49H
A048    FE 41                CP    41H
A04A    11 EC 00             LD    DE,00ECH
A04D    28 42                JR    Z,PTRNXT
A04F    FE 53                CP    53H
A051    11 D2 00             LD    DE,00D2H
A054    28 3B                JR    Z,PTRNXT
A056    FE 44                CP    44H
A058    11 BC 00             LD    DE,00BCH
A05B    28 34                JR    Z,PTRNXT
A05D    FE 46                CP    46H
A05F    11 AF 00             LD    DE,00AFH
A062    28 2D                JR    Z,PTRNXT
A064    FE 47                CP    47H
A066    11 9E 00             LD    DE,009EH
A069    28 26                JR    Z,PTRNXT
A06B    FE 48                CP    48H
A06D    11 8C 00             LD    DE,008CH
A070    28 1F                JR    Z,PTRNXT
A072    FE 4A                CP    4AH
A074    11 7E 00             LD    DE,007EH
A077    28 18                JR    Z,PTRNXT
A079    FE 4B                CP    4BH
A07B    11 76 00             LD    DE,0076H
A07E    28 11                JR    Z,PTRNXT
A080    FE 00                CP    00H
A082    28 07                JR    Z,RDEXT
A084    FE 03                CP    03H
A086    CA 00 00             JP    Z,0000H
A089    FB                   EI
A08A    C9                   RET
A08B    AF          RDEXT:   XOR   A
A08C    32 27 A0             LD    (CHNPTR+1),A
A08F    18 6F                JR    PSGL1
A091    3A 26 A0    PTRNXT:  LD    A,(CHNPTR)
A094    FE 00                CP    00H
A096    26 00                LD    H,00H
A098    28 08                JR    Z,PTRLB
A09A    FE 01                CP    01H
A09C    26 02                LD    H,02H
A09E    28 02                JR    Z,PTRLB
A0A0    26 04                LD    H,04H
A0A2    3C          PTRLB:   INC   A
A0A3    FE 03                CP    03H
A0A5    20 01                JR    NZ,PSGL2
A0A7    AF                   XOR   A
A0A8    32 26 A0    PSGL2:   LD    A,(CHNPTR)
A0AB    6B                   LD    L,E
A0AC    CD 28 A0             CALL  PSGSET
A0AF    24                   INC   H
A0B0    6A                   LD    L,D
A0B1    CD 28 A0             CALL  PSGSET
A0B4    3A 27 A0             LD    A,(CHNPTR+1)
A0B7    B7                   OR    A
A0B8    CA 50 A1             JP    Z,PSGL3
A0BB    3A 26 A0             LD    A,(CHNPTR)
A0BE    00                   NOP
A0BF    00                   NOP
A0C0    21 0F 08             LD    HL,080FH
A0C3    CD 28 A0             CALL  PSGSET
A0C6    21 00 09             LD    HL,0900H
A0C9    CD 28 A0             CALL  PSGSET
A0CC    21 0A 0A             LD    HL,0A0AH
A0CF    CD 28 A0             CALL  PSGSET
A0D2    FE 01                CP    01H
A0D4    28 28                JR    Z,PSGL4
A0D6    21 0A 08             LD    HL,080AH
A0D9    CD 28 A0             CALL  PSGSET
A0DC    21 0F 09             LD    HL,090FH
A0DF    CD 28 A0             CALL  PSGSET
A0E2    21 00 0A             LD    HL,0A00H
A0E5    CD 28 A0             CALL  PSGSET
```

```
A0E8    FE 02                   CP      02H
A0EA    28 12                   JR      Z,PSGL4
A0EC    21 00 08                LD      HL,0800H
A0EF    CD 28 A0                CALL    PSGSET
A0F2    21 0A 09                LD      HL,090AH
A0F5    CD 28 A0                CALL    PSGSET
A0F8    21 0F 0A                LD      HL,0A0FH
A0FB    CD 28 A0                CALL    PSGSET
A0FE    18 41           PSGL4:  JR      PSGL5
A100    3A 26 A0        PSGL1:  LD      A,(CHNPTR)
A103    21 0A 08                LD      HL,080AH
A106    CD 28 A0                CALL    PSGSET
A109    21 00 09                LD      HL,0900H
A10C    CD 28 A0                CALL    PSGSET
A10F    21 00 0A                LD      HL,0A00H
A112    CD 28 A0                CALL    PSGSET
A115    FE 01                   CP      01H
A117    28 28                   JR      Z,PSGL5
A119    21 00 08                LD      HL,0800H
A11C    CD 28 A0                CALL    PSGSET
A11F    21 0A 09                LD      HL,090AH
A122    CD 28 A0                CALL    PSGSET
A125    21 00 0A                LD      HL,0A00H
A128    CD 28 A0                CALL    PSGSET
A12B    FE 02                   CP      02H
A12D    28 12                   JR      Z,PSGL5
A12F    21 00 08                LD      HL,0800H
A132    CD 28 A0                CALL    PSGSET
A135    21 00 09                LD      HL,0900H
A138    CD 28 A0                CALL    PSGSET
A13B    21 0A 0A                LD      HL,0A0AH
A13E    CD 28 A0                CALL    PSGSET
A141    21 09 0D        PSGL5:  LD      HL,0D09H
A144    CD 28 A0                CALL    PSGSET
A147    FB                      EI
A148    C9                      RET
A149    00                      NOP
A14A    00                      NOP
A14B    00                      NOP
A14C    00                      NOP
A14D    00                      NOP
A14E    00                      NOP
A14F    00                      NOP
A150    3E 01           PSGL3:  LD      A,01H
A152    32 27 A0                LD      (CHNPTR+1),A
A155    3A 26 A0                LD      A,(CHNPTR)
A158    21 0F 08                LD      HL,080FH
A15B    CD 28 A0                CALL    PSGSET
A15E    21 00 09                LD      HL,0900H
A161    CD 28 A0                CALL    PSGSET
A164    21 0A 0A                LD      HL,0A0AH
A167    CD 28 A0                CALL    PSGSET
A16A    FE 01                   CP      01H
A16C    28 28                   JR      Z,PSGL6
A16E    21 0A 08                LD      HL,080AH
A171    CD 28 A0                CALL    PSGSET
A174    21 0F 09                LD      HL,090FH
A177    CD 28 A0                CALL    PSGSET
A17A    21 00 0A                LD      HL,0A00H
A17D    CD 28 A0                CALL    PSGSET
A180    FE 02                   CP      02H
A182    28 12                   JR      Z,PSGL6
A184    21 00 08                LD      HL,0800H
A187    CD 28 A0                CALL    PSGSET
A18A    21 0A 09                LD      HL,090AH
A18D    CD 28 A0                CALL    PSGSET
A190    21 0F 0A                LD      HL,0A0FH
A193    CD 28 A0                CALL    PSGSET
A196    FB              PSGL6:  EI
A197    C9                      RET
```

Ⅳ X1ゆたかな周辺機器と拡張性

●

X1システムの拡張方法
フロッピディスク
プリンタインタフェース
漢字ROM
より進んだシステムへ

X1システムの拡張方法

■システムの構成

X1本体を使いこなしてくると「漢字プリンタがあれば……」「ビデオの編集を……」と，システムを拡張したくなります。X1シリーズはこれらの要求に応じる豊富なラインナップを提供してくれます。図IV-1は標準的な周辺機器の接続関係を表したものです。

ディスプレイは，それぞれのタイプに対応したものが用意されています。一般のRGBカラーディスプレイも使えますが，パソコンテレビとしての機能はなくなってしまいます。

たとえばコンピュータ画像とカラーテレビの映像を，同じブラウン管上で重ね合わせるスーパーインポーズ機能もなくなります(X1ターボでは可能です)。Hu-BASICのコマンドにある種々のテレビのコントロールが不可能ですし，VTRを使った応用もできません。ただし，市販ソフトでこの機能を生かしたものは少ないので，実用上はそれほど不都合ではないかもしれません。

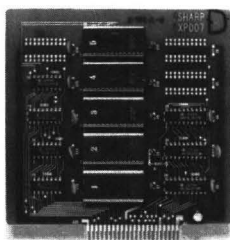
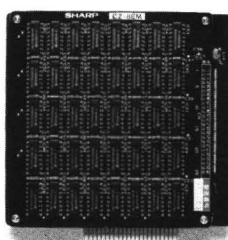
プリンタは，セントロニクス準拠のポートが装備されていますので，一応セントロニクス対応のタイプならどれでも接続できるはずですが，タイミングが厳密に統一されていないのでうまく動作しない場合も考えられます。プリンタが純正品でないとX1独自のキャラクタが印字できないことがあるので，X1用と明示されたものが適当でしょう。

X1, X1Cにミニフロッピディスクを装備するときには，拡張I/Oポートあるいはボックスを接続する必要があります。これにディスクインタフェース基板を差し込めば，ディスク本体を接続することができ，これによってX1およびX1CはX1独自の機能も持ち合わせた，CP/Mの走るマシンとなります。

X1 turboには，拡張I/Oポートが最初から装備されています(フリースロットは2個)。

拡張I/Oポート(ボックス)用に次のような興味深いインタフェースが発売されています(Photo 1～6)。

RS-232Cカード： 他のコンピュータや端末との交信をする。

PhotoⅣ-1
ROM　BASICカード
CZ-8RB

PhotoⅣ-2
320KB外部メモリ
CZ-8EM

PhotoⅣ-3
フロッピディスクインタフェース
CZ-8FA

図VI-1 X1システム構成図

X1turbo(CZ-852C)
CZ-812CE/R＊

X1F(CZ-811CE/R)
(CZ-812CE/R)
CZ-811DE/R＊

X1CS(CZ-803C)
CZ-801D＊

X1CK(CZ-804C)
CZ801D＊

X1D(CZ-802C)
CZ802D＊

X1(CZ-800C)
CZ-800D＊

カラーディスプレイテレビ
グリーンディスプレイ
家庭用カラーテレビ
RFビデオコンバータCZ-8VC

ミニフロッピディスクドライブ CZ-801F (S/R)
① フロッピディスクインタフェース CZ-8B01
コンパクトフロッピディスクドライブCZ-300F(S/R)
② 増設用コンパクトフロッピディスクドライブ CZ-31F(S/R)
③ ディスクBASIC CZ-8FB01
CP/M CZ-5CPM
ディスクBASIC CZ-8W301
CP/M CZ-3CPM
コンパクトフロッピディスクCZ-3FBD

ドットプリンタ CZ-8PD2(S/R)
ドットプリンタ CZ-800P
漢字プリンタ CZ-80PK(S/R)
熱転写プリンタ CZ-8PN1
カラープロッタプリンタ CZ-8PP2(S/R)

④ 拡張I/OポートCZ-8EP
ROM BASICカード CZ-8RB
ユニバーサルI/Oカード CZ-8UI
RS-232Cカード CZ-8RS
320KB 外部メモリ CZ-8EM
⑥ 漢字ROM CZ-8KR
⑦ グラフィックRAM CZ-8GR
ユーザー拡張外部機器
拡張I/Oボード CZ-8BE1
⑤ 拡張I/OボックスCZ-81EB(S/R)

デジタルテロッパCZ-8DT
ビデオマルチテロッパ
パーソナルテロッパ
ビデオカセットレコーダ
ビデオディスクプレーヤ
ビデオ入出力端子付カラーテレビ
ポータブルビデオカセットレコーダ
ビデオカメラ
ビデオディスクシステム
⑧ カセットデータレコーダ(市販品)
ジョイスティック(市販品)

＊は対応ディスプレイ

＊ディスプレイテレビ(CZ-800D、801D、802D)はX1シリーズのパーソナルコンピュータすべてに組み合わせ可能

①CZ-300Fをパーソナルコンピュータ(CZ-800C、CZ-803C、CZ-804C)に接続する場合、必要となります
②パーソナルコンピュータ(CZ-802C)の増設用と共用です
③ディスクBASICはフロッピディスクインタフェースとともにCZ-801Fに同梱されています
④パーソナルコンピュータ(CZ-800C、CZ-802C)を拡張する場合、必要となります
⑤パーソナルコンピュータ(CZ-803C、CZ-804C)で3ポート以上必要な場合に使用できます。その際には拡張I/Oボード(CZ-8BE1)が必要となります
⑥パーソナルコンピータ(CZ-804C)には、漢字ROMCZ-8KRと同性能のものが標準実装されています
⑦パーソナルコンピュータ(CZ-802C、CZ-803C、CZ-804C)には標準実装されています
⑧パーソナルコンピュータ(CZ-802C)に使用できます

Photo IV-4

RS-232Cカード
CZ-8RS

Photo IV-5

ユニバーサルI/Oカード
CZ-8UI

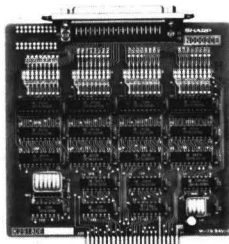
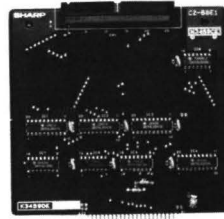



Photo IV-6

拡張I/Oボード
CZ-8BE1

ユニバーサルI/Oカード：種々の外部機器とのデータ交換をする。

320KB外部メモリ：大容量外部RAMとして使う。

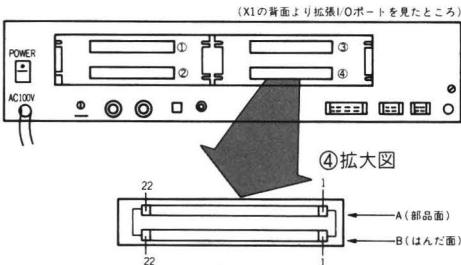
漢字ROM：画面へ漢字を表示する(X1 turbo以外)。

ROM BASICカード：電源ON後，すぐBASICが起動する。

このほかにも，自作のインタフェースを作るためのX1用ユニバーサルボードがI/Oデータ機器社などから発売されています。

■拡張I/Oポートの使い方

X1(C/D/turboを除く)の場合，ディスクのインタフェース基板を差し込むためには拡張I/Oポートが必要です。このポートには4つまでインタフェース基板が差し込めます(図Ⅳ-2)。I/Oポートに出ている信号の意味を理解してインタフェースを作り独自のシス

図Ⅳ-2　X1背面拡張I/Oポート

図Ⅳ-3　拡張I/Oポート信号名

B(はんだ面)		A(部品面)
+5V	1	+5V
DB3	2	DB2
DB4	3	DB1
DB5	4	DB0
DB6	5	GND
DB7	6	AB15
CPU CLOCK	7	AB14
$\overline{\text{M1}}$	8	AB13
$\overline{\text{WR}}$	9	AB12
$\overline{\text{RD}}$	10	AB11
$\overline{\text{IORQ}}$	11	AB10
$\overline{\text{MREQ}}$	12	AB9
GND	13	AB8
$\overline{\text{HALT}}$	14	AB7
IEI(1~4)	15	AB6
IEO(1~4)	16	AB5
RESET	17	AB4
$\overline{\text{EXIO}}$	18	AB3
$\overline{\text{EXINT}}$	19	AB2
$\overline{\text{EXWAIT}}$	20	AB1
$\overline{\text{NMI}}$	21	AB0
GND	22	GND

図Ⅳ-4　X1/X1 turboの相異点

	X1	X1turbo
B-14	$\overline{\text{HALT}}$	$\overline{\text{BUSAK}}$
B-21	$\overline{\text{NMI}}$	$\overline{\text{EX-RDY}}$

注）turboでは$\overline{\text{HALT}}$と$\overline{\text{NMI}}$はメイン基板上に2ピンコネクタが付いています。

図Ⅳ-31　拡張I/Oポート信号の詳細

信号名	端子番号	方向(入力←出力→)	信号内容
AB0~AB15	A6~21	→	16ビットアドレスバス
DB0~DB7	A2~4, B2~6	↔	8ビットデータバス
$\overline{\text{MREQ}}$	B12	→	メモリアクセス要求
$\overline{\text{IORQ}}$	B11	→	I/Oアクセス要求
$\overline{\text{RD}}$	B10	→	リードストローブ
$\overline{\text{WR}}$	B9	→	ライトストローブ
CPU CLOCK	B7	→	4MHzの単相クロック
$\overline{\text{EXIO}}$	B18	→	I/Oアドレスデコード(0000H~0FFFH)
$\overline{\text{EXINT}}$	B19	←	外部機器からの割り込み要求
$\overline{\text{EXWAIT}}$	B20	←	外部機器からのウェイト
$\overline{\text{M1}}$	B8	→	OPコードフェッチサイクル
$\overline{\text{NMI}}$	B21	→	外部機器からのノンマスカラブル割り込み要求
IEI(1~4)	B15	→	割り込みイネーブル入力(外部機器の)
IEO(1~4)	B16	←	割り込みイネーブル出力(外部機器の)
$\overline{\text{HALT}}$	B14	→	CPU HALT状態
RESET	B17	→	リセット信号

テムを作り上げていくことは，用意されたものを使っていたのでは得られない魅力があります。

コネクタの信号は図IV-3,4に示します。A，Bの区別，1～22の方向は間違えないようにしてください。それぞれの信号の意味も示しておきます(図IV-5)。

X1は，64Kバイトもの広いI/O空間を持っており，しかもシングルアクセスモードと同時アクセスモードのふたつのモードの切り換えができます。

ユーザーに開放されているI/O空間はシングルアクセスモードにおける0000H～0FFFHです。そしてアドレス16ビットのデコードやモードの切り換えの手間を省くために，$\overline{\mathrm{EXIO}}$信号が用意されています。

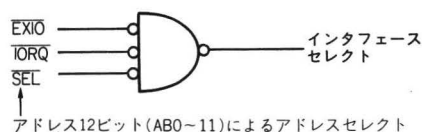
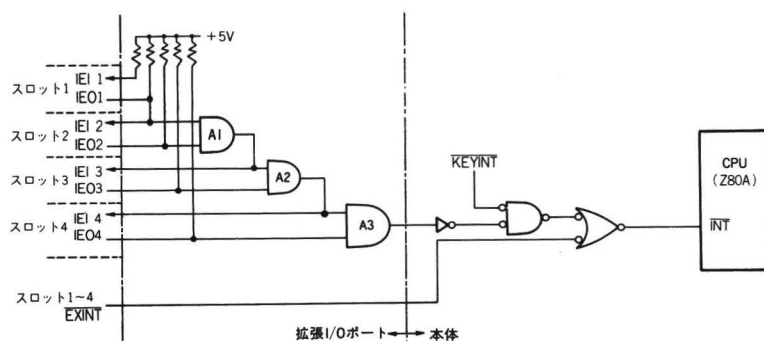
この信号はアドレスバスの上位4ビット(AB12～AB15)がすべて0で，かつシングルアクセスモードのときのみ0となります。したがって，インタフェースのセレクト信号は図IV-6のように作ることができます。

なおユーザーI/O空間の中で0100H～0FFFHについては，シャープで開発する周辺機器のインタフェースに予約されており，使用すると重複する可能性があるので，0000H～00FFHを選択するようにしてください(図IV-7)。

インタフェースを設計するときぜひ考えなければならないのは「割り込み」です。CPUが何かプログラムを実行しているときに，いきなり外部機器から待ったがかかりその処理をしなければならないことがあるものです，また複数の割り込み要因が発生することもあります。

図IV-6　インタフェースセレクト信号の作成

図IV-7　周辺機器のI/Oポート

名目インタフェース	I/Oポート
RS-232C (CZ-8RS)	0C＊0～0C＊7
320KRAM	0D00～0D07
BASIC ROMボード (CZ-8RB01)	0E00～0E03
漢字ROMボード (CZ-8KR)	0E80～0E82
フロッピディスクインタフェース (CZ-8 FA)	0FF8～0FFC

図IV-8　X1におけるデイジーチェインによる割り込み制御

そこでX1では，4つのポートからの割り込みと割り込みキー入力の順位を，シンプルな「デイジーチェイン」によって，決定しており優先順位は次のようになっています。

X1… スロット1＞スロット2＞スロット3＞スロット4＞割り込みキー入力

X1turbo… スロット1＞スロット2＞SIO＞DMA＞CTC＞割り込みキー入力

このX1の割り込み制御は図IV-8のような回路で実現しています。

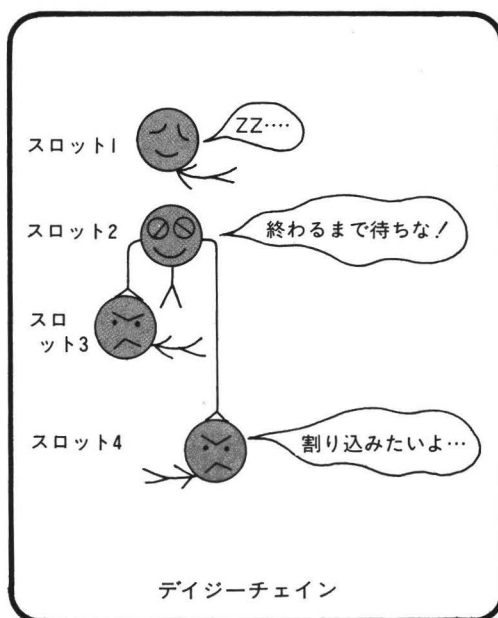
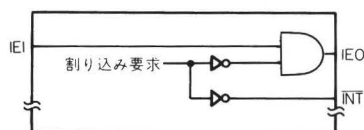
スロット3を例にとって説明しましょう。

割り込みをかけようとするとき，IEI3の信号を調べ，スロット3より優先順位の高いスロット2が割り込んでいると，IEO2＝0なのでA1のANDゲートの出力IEI3が0となり，割り込みはかかりません。

反対に，スロット1もスロット2も割り込んでいないとき，つまりIEO1＝IEO2＝1のときは，A1の出力が1となり，IEI3＝1となって，スロット3による割り込みが発生します。さらにIEO3＝0となり，IEI4＝0になるためスロット4の割り込みは押えられます。

これが「デイジーチェイン」の仕組みです。

Z80系のインタフェースLSIには，IEIやIEOの端子があるのが普通です。これは，図IV-9のような論理回路がLSI内にあると考えてもよいでしょう。

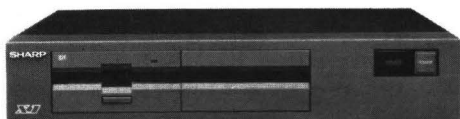
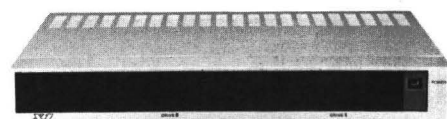
図IV-9　IEI, IEO信号の仕組み

フロッピディスク

■フロッピディスクの構造

フロッピディスクドライブは，パーソナルコンピュータの外部記憶を行う装置のひとつで，比較的大量のデータを高速処理することができます。記憶媒体としてフロッピディスケット(ディスク)を使います。

PhotoⅣ-7 コンパクトフロッピディスクドライブCZ-300F

PhotoⅣ-8 ミニフロッピディスクドライブ CZ-800F

フロッピディスケットは，ポリエステルの薄い円盤の表面に磁気材料をコーティングし，これをジャケットと呼ばれる正方形の塩化ビニールのパッケージに納めたものです。直径8インチ，5.25インチのものなどがあります。

ディスケットの表面に，ディスクドライブによって磁気的にデータを記録し，記憶メディアとして読み書き保存をします。フロッピディスケットには次のような特徴があります。

・記憶容量が大きい

・ランダムアクセスが可能

・データ転送速度が速い

カセットテープは，データの読み書きが順番，つまりシリアルアクセスしかできません。これに対し，ディスクは任意の位置へいって読み書きできるランダムアクセスがきくのでアクセスが速いのです。

フロッピディスケットは，大きさや記録方式がいろいろあり，ディスクドライブもそれに応じて多くの種類があります。

最初に出現したディスケットはIBM社が1972年に発表した8インチの標準ディスクで，これは片面単密度の物理的記録方式のものです。このIBMの記録方式がその後の8インチディスケットの標準フォーマット，いわゆる「IBMフォーマット」として他機種の8インチディスクドライブ間の互換性に大きく貢献しました。以後8インチ以外のディスケットが各社各様の仕様で開発，商品化されましたが，8インチのように互換性が統一されているとはいいがたい状況です。サイズには5.25インチ，4インチ，3.5インチ，3インチなどがあります。また物理的記録方式をみると，5.25インチの場合，片面倍密度(Single Sided Double Density, 1D)，両面倍密度(Double Sided Double Density, 2D)，両面倍密度倍トラック(2DD)，両面高密度倍トラック(2HD)などがあります。

ディスケットの記録方式も種々あります。単密度のものはFM(Frequency Modulation)方式でリード/ライトを行います。倍密度のものはMFM(Modulated Frequency Modulation)方式，またはM^2FM(Modified Modulated Frequency Modulation)方式を採用して

います。

5.25インチディスクの構造例

フロッピディスクの構造を，5.25インチのものを例にとり説明します(図IV-10)。

センターホール：ディスケットの中央にある大きな穴です。ディスクドライブに差し込んだとき，スピンドルに固定するためのものです。

column 7

記録方式

現在，一般に行われているディスク記録方式は，①FM記録方式，②MFM記録方式，③M²FM記録方式の3つがあります。

FM記録方式

単密度記録方式と呼ばれます。

i．データ信号が1のとき，ビットセルの中央にデータビットを書く。0のときは書かない。

ii．各ビットセルの先頭に，クロックビットを書く。

MFM記録方式

倍密度記録方式でよく用いられる方式です，X1シリーズのCZ800Fもこの方式をとっています。

i．データ信号が1のとき，ビットセルの中央にデータビットを書く。0のときは書かない。

ii．直前および現在のビットセルのデータ信号が両方とも0のとき，現在のビットセルの先頭にクロックビットを書く。

この方式は，ビット密度がFM方式に比べて約1/2ですむため，記録容量が倍とれます。

M²FM記録方式

MFM方式をさらに変調したものです。シュガート社が開発したが，現在ではあまり使われていません。

i．データ信号が1のとき，ビットセルの中央にデータビットを書く。0のときは書かない。

ii．直前および現在のビットセルのデータ信号が両方とも0のとき，直前のビットセルにクロックビットがないとき，現在のビットセルの先頭にクロックビットを書く。

この方式もビット密度がFM方式の約1/2なので，MFM方式と同様にFM方式の2倍の記録容量が得られます。

ヘッドウィンドウ：細長い穴の部分でディスクドライブのリード/ライトヘッドがディスケットと接して読み書きを行うためのものです。

インデックスウィンドウ：一番小さい丸い穴のことで，ディスケットの回転位置の検出のためのものです。

ライトプロテクトノッチ：ディスケットの中身を保護する場合，ここにライトプロテクトシールを貼る部分です。ライトプロテクトシールの有無は光学的に検出されます。

ディスケットへの記録は，ヘッドウィンドウを通し回転している同心円上に行われます。

市販のディスケットは，そのままではポリエステル円盤に磁気材料をコーティングしただけですから，使用する前にフォーマットしなければなりません。

フォーマットというのは，ディスク上の領域を磁気的に区分して，記録するとき任意の場所が参照できるようにすることです。

5.25インチ両面倍密度の場合は，まず中心から外に向かって40の同心円の区分を作ります。この各区分をトラックといい，いちばん外側をトラック0として中心方向に数え，いちばん内側がトラック39となります。

次にディスクを放射状に16等分します。この16等分された各トラックをセクタといいます(図Ⅳ-11)。5.25インチ両面倍密度では，

1セクタ＝256バイト

のフォーマット方式となっています。

図Ⅳ-10　フロッピディスクの構造

図Ⅳ-11　トラックとセクタ

図Ⅳ-12　外部ファイル単位相関

ドライブ	トラック(クラスタ)	セクタ(レコード)	バイト	ビット
1	40	640	163,840	1,310,720
	1	16	4,096	32,768
		1	256	2,048
			1	8

1．256バイトを1レコードとよびますが，フロッピディスクの場合はとくに1セクタとよびます。
2．レコード，クラスタ（後述），トラック，ドライブの数え方は0からはじまります。
　セクタは1番からはじまります。

このフォーマットはディスクの両面に対して行われます。したがって1枚のディスクは，

2 × 40 × 16 × 256 ＝327680バイト/ディスケット

〔サイド/ディスケット〕〔トラック/サイド〕〔セクタ/トラック〕〔バイト/セクタ〕

の記録ができ，1Kバイト＝1024バイトですから，320Kバイト/ディスケットになります。

ディスクをアクセスするとき，ディスケット上の位置を指定するのにトラックとセクタのふたつの値を使い極座標形式で位置を検出します。このとき位置の基準になるのが前述のインデックスホールという穴で，これによって各点の絶対的位置が決まります。

ところでX1はフロッピディスクドライブのほか，グラフィックV-RAMなどを外部ファイルとして使用することができます。扱う単位の関係は図IV-12のとおりです。

■フロッピディスクドライブ

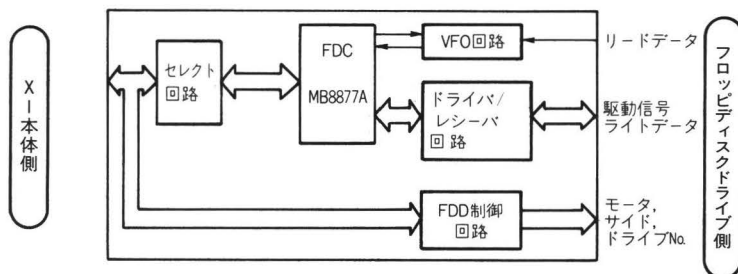
フロッピディスクドライブは，通常パソコンに接続して制御されます。つまりディスクドライブはパソコンのコマンドによって，フロッピディスクへの書き込みと読み出しを行います。

X1用のフロッピディスクドライブユニットCZ-800F(CZ-801Fも共通，仕様・図IV-13/インタフェース構成・図IV-14)は，両面倍密度の5.25インチドライブを2基装備しています。X1本体とはインタフェース基板(図IV-15)CZ-8FAを介して接続し，4ドライブ(2ユニット)まで接続でき，フロッピディスクへのデータの記録方式はMFM方式です。X1本体とのデータ転送は256バイト(1セクタ)単位で行います。

図IV-13　CZ-800Fの仕様

記録方式		両面倍密度記録(MFM)	記録方式		両面倍密度記録(MFM)
記録容量	アンフォーマット時	1　MByte	情報転送速度		250KByte/Sec
(ユニット当たり)	フォーマット時(16セクタ/トラック)	655.36KByte	平均回転待時間		100　msec
記録密度		5876　bpi	アクセス時間	平均アクセス時間	281　msec
トラック密度		48　TPI		トラック間	20　msec
シリンダ総数		40		セトリング時間	15　msec
トラック総数		80	ヘッドロード時間		50　msec
媒体回転速度		300　RPM	モータ起動時間		1　sec

MFM記録方式(Modified Frequency Modulation)：倍密度記録方式
bpi(bit per inch)：1インチ当たり何ビット記憶できるかを表す単位
TPI(Tracks Per Inch)：1インチ当り何トラックあるか表す単位。トラック・ピッチは48TPIで0.529mm
RPM(Revolution Per Minute)：毎分回転数

図IV-14　フロッピディスクインタフェース基板CZ-8FAブロック構成図

■フロッピディスクコントローラ(FDC)

フロッピディスクドライブのインタフェースはほかの周辺機器と異なり，高速パルス信号やレベル信号などを使うため制御がかなり複雑です。基本TTLでコントロール回路を組もうとすると設計が難しく，回路も複雑になり多くの部品が必要です。部品が多いほど動作チェックの時間や費用もかかります。

FDCは，これらの制御機能を持つ専用LSIです。LSI化することによって設計が簡単になるほか，部品の数が減りコストパフォーマンスを高めることにつながります。

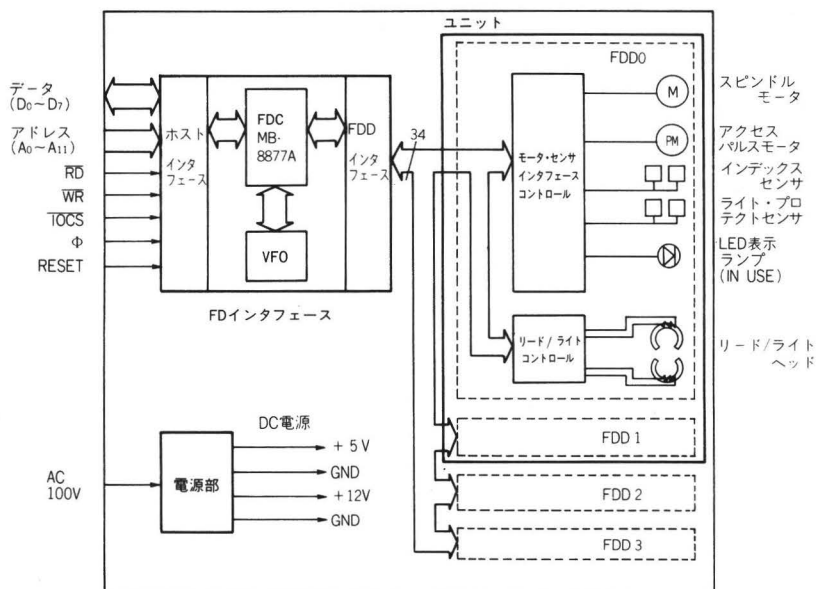
FDCの内部レジスタに必要データをCPUから送り込みコマンドを与えれば，初期化，シーク，リード，ライトなどのフロッピディスクドライブの各種動作を自動的に行います。

FDCは各社からいろいろな仕様のものが出ていますが，X1は富士通製のMB8877Aを採用しています。このほかよく使われるFDCとしては，NECのμPD765Aがあります。

FDC MB8877A

MB8877A・FDCは，フロッピディスクドライブ(FDD)を制御するための1チップコントローラLSIで，ホストコンピュータからのソフトウェアで処理されます。おもな働きは次のとおりです。

1. データの変調と復調：フロッピディスクにCPUからのデータを変調して記録し，フロッピディスクのデータを復調してCPUに出力する。
2. FDD駆動信号の発生と制御：ヘッドの動作など，FDDのメカニズムの制御を行う。

図Ⅳ-15　CZ-800Fインタフェース構成ブロック図 (CZ-8FA)

FDC(Floppy Disk Controller)：フロッピディスク制御装置

VFO(Variable Frequency Oscillator)：可変周波数発振器。MFM記録方式では，クロックビットをたよりにデータビット用のウィンドウを発生させる方法が不可能となる。そこでVFOでは，フロッピディスクからのリード データ中にクロックビットがなくても，データビット用ウィンドウと，クロックビット用のウィンドウを正確に発生させる役目を受け持つ

FDD(Floppy Disk Drive)：フロッピディスク装置

3．FDD状態検出信号による制御：FDDの状態を検出して，ソフト，ハードの制御を行う。

4．ホストコンピュータとのインタフェース：CPUからのFDDの制御やライトデータまたはFDDからCPUへのリードデータのインタフェースを行う。

フロッピディスクコントローラの制御

X1は，FDCの制御に図IV-16のI/Oポートを割り当てています。

MB8877A FDCの内部には，①コマンド，②ステータス，③トラック，④セクタ，⑤データ⑥データシフトのレジスタがあります。これらのレジスタに必要なデータやコマンドを与えたり，入力することによってFDDをコントロールします。またステータスを知ることができます。

ただし，次の3つのコントロールはMB8877A FDCで処理できません。

図IV-16　FDC制御I/Oポート

ポートアドレス	(X1本体からみて)入/出力	内　　容
0FF8H	IN	ステータスレジスタ
	OUT	コマンドレジスタ
0FF9H	IN / OUT	トラックレジスタ
0FFAH	IN / OUT	セクタレジスタ
0FFBH	IN / OUT	データレジスタ
0FFCH	OUT	ドライブセレクト，ディスクサイド，モータオンレジスタ

column8

X1の各機種とフロッピディスク

本書ではフロッピディスクとして両面倍密度のミニフロッピディスクCZ-800(801) Fを想定して話をしています。しかしX1Dに内蔵(オプションとしてはCZ-300F)されているコンパクトフロッピディスクもまったく同様なコントロールができるので，そのまま適用することができます。またX1 turbo内蔵のミニフロッピディスクにも適用が可能です。

X1 turboは記録容量を次の3種類から選ぶことができます。

記録方式	記録容量	
	アンフォーマット時	フォーマット時
2D：両面倍密度	500KB	320KB
2DD：両面倍密度 倍トラック	1MB	640KB
2HD：両面高密度	1.6MB	1MB

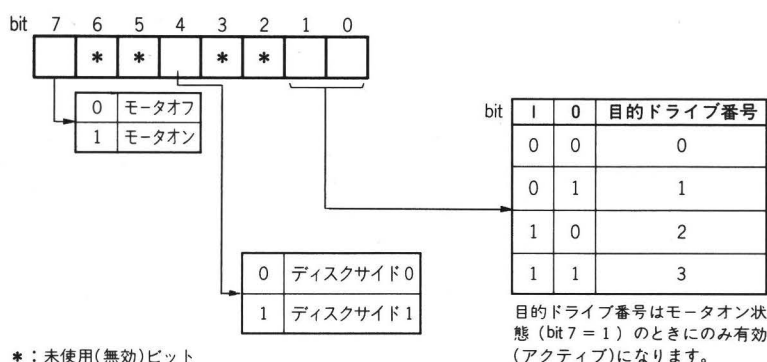
注　従来のX1シリーズでは2D方式しかサポートしていない

X1 turboでは，ミニフロッピだけでなく8インチも接続できるように考慮されており，さらにハードディスクや2DD,2HDディスクも自動切り替えが使える設計になっています。

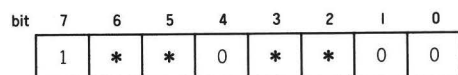
・ドライブ番号(0～3)の選択
・モータのON/OFFの制御
・ディスクサイドの選択

X1では，これらの制御は直接CPUからI/Oポート0FFCHを介して行います。

なおドライブ番号，ディスクサイド，モータオン/オフの設定I/Oポート0FFCHのビット構成は図IV-17のとおりです。

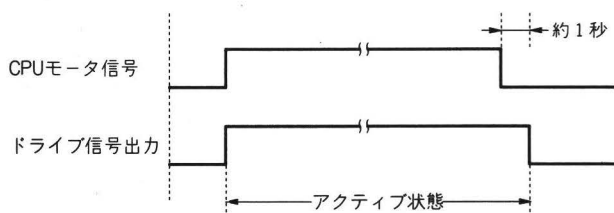
図IV-17　0FFCH I/Oポートビット構成

モータオン，ディスクサイド0，ドライブ0の場合は，図IV-18のように指定します。

図IV-18　0FFCH I/Oポート設定例

bit	7	6	5	4	3	2	1	0
	1	＊	＊	0	＊	＊	0	0

＊のところは無効ビットで0にしても1にしても関係ありません。

リード/ライトヘッドは，ドライブ信号によってロード，アンロードしています。このため連続したコマンドを実行する場合(たとえばリードからライト動作)，ひとつのコマンド終了ごとにドライブ信号をすぐノンアクティブにしているとヘッドのタップが発生します。

これを防ぐためにCPUからのモータオフ信号が出力されてからも，約1秒間ドライブ信号がアクティブ状態にあるように設計されています(図IV-19)。この1秒間に次のコマンドが入力されれば，モータ起動時間やヘッドのタップがなくなります。

図IV-19　モータ信号とドライブ信号タイミング図

■ディスクドライブの動作

ディスクドライブの動作は，①ドライブセレクト，②モータ，③リード/ライトヘッドのロードの3つに分けられます。

ドライブ セレクト： X1では，インタフェース基板を介してFDDを4台まで接続できるので，どのドライブを実行するのか指定しなければなりません。モータオン期間(後述)に，ドライブセレクトレジスタによって選択されたドライブが有効となります。

column 9

外部デバイスファイルの取り扱い

X1 BASICではディスクに限らず外部デバイスを使うときは，デバイスの違いに関係なく統一処理ができるように，外部デバイスの位置指定に，論理的にレコードという単位を用いています。

各デバイスの先頭の値をレコード番号0とし，256バイト単位にレコード番号をふりつけます。レコード番号の使用で，各デバイスに対する統一処理が可能となりました。デバイスの区別は，ファイルディスクリプタを使って行います。

各デバイスのレコード番号と，物理的アドレスの関係は次の表のとおりです。

デバイスレコード番号と物理的アドレス相関表

レコード番号	フロッピディスク	外部メモリ EMM0：～EMM9：	グラフィックメモリ MEM：
0000H	サイド0，トラック0，セクタ1	アドレス＝000000H	IOアドレス＝4000H
0001H	サイド0，トラック0，セクタ2	アドレス＝000100H	IOアドレス＝4100H
⋮	⋮	⋮	⋮
0010H	サイド1，トラック0，セクタ1	アドレス＝001000H	IOアドレス＝5000H
⋮	⋮	⋮	⋮
00BFH	サイド1，トラック5，セクタ16	アドレス＝00BF00H	IOアドレス＝FF00H
⋮	⋮	⋮	
04FFH	サイド1，トラック39，セクタ16	アドレス＝04FF00H	

これらの各デバイスには，ファイルを管理するふたつの領域があり，INITコマンドで初期設定されます。このふたつの管理領域をFAT(File Allocation Table)とDIR (Directory：ディレクトリ領域) と呼びます。

FATは，レコード番号14の位置にあり，ファイルの格納状況を記録，管理するところです。FATの大きさは，デバイスの種類 (記憶空間の大きさ) によって違います。

フロッピディスクの外部メモリの場合は，00H～4FHまでの80バイト (80×4K＝320K)

グラフィックメモリの場合は00H～0BHまでの12バイト (12×4K＝48K) が使われます。

FATの各バイトは，デバイス内の各クラスタに対応し，各バイトの内容が，ファイルのつながりとクラスタの使用状況などを表します。初期設定のとき，FATの各バイトに8FHが書き込まれます。各バイトの内容は01H～8FHの値を持ち，次のような意味を含んでいます。

01H～7FH：ファイルが格納される次のクラスタ番号を示す。

モータ：ディスクを回転させるモータの駆動のことです。連続的にコマンドを実行する場合その約1秒前からモータは回転し，最終のコマンドが終了してからも約1秒間モータは回転し続けます。

モータリード/ライトヘッドのロード：ドライブがセレクトされると，そのドライブのリード/ライトヘッドがロードされます。コマンドが終了してから約1秒間後にリード/ライトヘッドはアンロード(ディスクのヘッドウィンドウから離れること)されます。

実際にFDDを動かしてみましょう。手順は次のとおりです。

80H～8FH：このクラスタでファイルが終了していることを示す。

この場合この値から7FHを引いた値がそのクラスタ中で使用されているレコード数を表します。1クラスタ＝16レコード＝4Kバイトです。

ディレクトリ領域は，レコード番号16～31の位置にあり，ファイルのモード，名前，メモリの格納（ロード）アドレス，作成年月日時分，ファイル格納先頭クラスタなどを管理するところです。ひとつのファイル当り32バイトが使用され，全部で128のファイルが記録できます。32バイトの構成は次のとおりです。

ファイルディスクリプタと入出力デバイス表

ファイルディスクリプタ	入出力デバイス名	用途	
		入力	出力
KEY：	キーボード	0	
CRT：	ディスプレイ		0
SCR：	ディスプレイ		0
LPT：	プリンタ		0
CAS：	カセット	0	0
0： ～ 3：	ディスクドライブ0 ～ ディスクドライブ3	0	0
MEM：	グラフィックメモリ	0	0
EMM0： ～ EMM9：	外部メモリ0 ～ 外部メモリ9	0	0

ファイルコントロールブロックの構成表

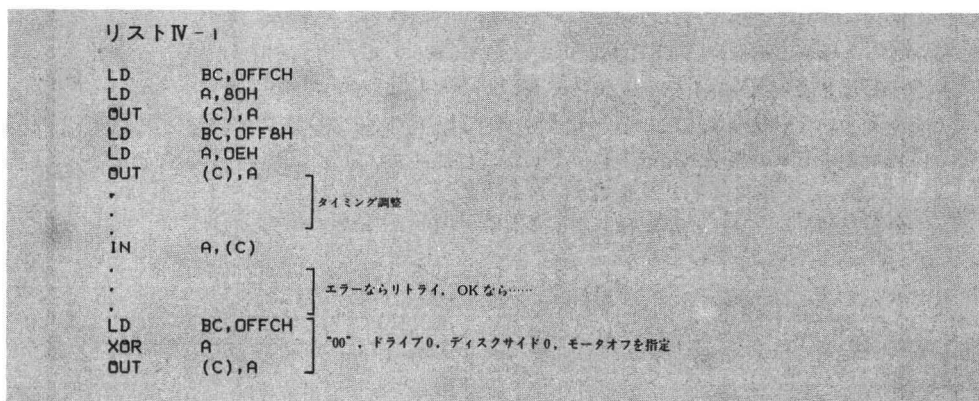
バイト	内容
00H	ファイルモード（種類）を表す：00HはKILLされたファイル FFHは使用ディレクトリ領域の終わり bit 0が1……Binファイル（機械語ファイル） bit 1が1……Basファイル（BASICテキストファイル） bit 2が1……Ascファイル（ASCIIセーブされたファイル） bit 3が1……SAVE" "　Pで秘密化されたファイル bit 4が1……FILESコマンドで表示しない 0……FILESコマンドで表示する bit 5が1……リードアフターライトON 0……リードアフターライトOFF bit 6が1……書き込み禁止ファイル 0……書き込み可能ファイル bit 7が1……Dir名(階層化ディレクトリのディレクトリ名) bit 0～2および7は同時に1になることは許されない
01H～0DH	ファイル名（13文字）
0EH～10H	拡張子（3文字）
11H	パスワード
12H，13H	ファイルの長さ（バイト数）(Bas及びObjのみ有効) L，Hの順に格納
14H，15H	ファイルのメインメモリ先頭アドレス（Objファイルのみ有効） L，Hの順に格納
16H，17H	ファイルのメインメモリ実行アドレス（Objファイルのみ有効） L，Hの順に格納
18H～1CH	作成年月日時分 18　19　1A　1B　1C 年　月 曜日　日　時　分 年，日，時，分：BCD表記 月，曜日　：BINARY表記
1DH～1FH	ファイル格納先頭クラスタ（1EHバイト），1DH，1FHバイトは，常に00H

これはIPLのところで説明したIPLロード用インフォメーションブロックとほぼ同じです。異なるのは1DH～1FHバイトで，インフォメーションブロックの場合は，ファイルの格納する先頭レコード番号を示すのに対し，ディレクトリの場合は，ファイルの格納する先頭クラスタ番号を示していることです。

1. I/O ポート 0FFCH を通して FDD を直接制御する信号(ドライブ番号，ディスクサイド番号，モータ制御)を出力します。
2. I/O ポート 0FF8H～0FFBH を通してコマンドコードや必要なデータを FDC に出力します。FDC はコマンドコードを解析して，各コマンドの内部ルーチンの動作に入ります。そして FDC の内部ルーチンによって FDD に駆動信号を入力します。次に主な動作について説明します。

リストア：リストア動作は，ディスクヘッドをトラック 0 に移動することを行います(リストIV-1)。コマンドレジスタ(I/O ポート 0FF8H)にリストア命令 0EH を出力します。出力後，ステータスレジスタ(I/O ポート 0FF8H)を読み込み，それが 04H であれば正常です。

リストIV-1

```
LD      BC,0FFCH
LD      A,80H
OUT     (C),A
LD      BC,0FF8H
LD      A,0EH
OUT     (C),A    ┐
.                │タイミング調整
.                ┘
.
IN      A,(C)
.                ┐
.                │エラーならリトライ，OKなら……
.                ┘
LD      BC,0FFCH ┐
XOR     A        │"00"，ドライブ0，ディスクサイド0，モータオフを指定
OUT     (C),A    ┘
```

シーク：シーク動作は，ディスクヘッドを任意のトラックへ移動することを行います。1EH のシークコマンドを FDC に出力します。すると内部ルーチンで，目的トラック番号(データレジスタ)と，現在リード/ライトヘッドのあるトラック番号(トラックレジスタ)とを比較し，ディスク駆動信号として DIRECTION 信号と STEP パルスを出力して，ディスクヘッドを目的トラックへ移動します。

シーク動作のとき 1EH のシークコマンドを FDC のコマンドレジスタ(I/O ポート 0FF8H)に出力する前に目的トラック番号を FDC のデータレジスタ(I/O ポート 0FFBH)に出力する。

現在のヘッドのトラック番号を FDC のトラックレジスタ（I/O ポート 0FF9H) に出力する必要があります(リストIV-2)。

リード／ライト：ディスクのリード/ライト動作は，前述のように 256 バイト(1 セクタ)単位で行われるので，ソフトウェアでも 1 セクタ単位でリード/ライトルーチンを用意し，必要なセクタ数のデータをリード/ライトします。手順は次のとおりです。

① FDD がリード/ライト可能かどうかを調べる。インタフェース基板の有無を判定する。ドライブセレクト，ディスクサイド，モータオンレジスタを設定する(I/O ポート 0FFCH)。FDD からの READY 信号を調べる。

② リード/ライトの目的トラック番号へディスクヘッドを移動させる(シーク動作)。
目的トラック番号を，FDC のデータレジスタへ格納する(I/O ポート 0FFBH)。
現在ヘッドのあるトラック番号を，FDC のトラックレジスタに格納する(I/O ポート

リストIV-2

```
LD      BC,0FFCH    ┐
LD      A,80H       │ドライブ0，ディスクサイド0，モータオンに設定
OUT     (C),A       ┘
LD      BC,0FFBH    ┐
LD      A,nnH       │目的トラックNO (nnH) をデータレジスタに設定
OUT     (C),A       ┘
LD      BC,0FF9H    ┐
LD      A,(TRKWK)   │①
OUT     (C),A       ┘
LD      BC,0FF8H    ┐
LD      A,1EH       │シークコマンド (1EH) をコマンドレジスタに設定
OUT     (C),A       ┘
.                   ┐
.                   │delay
.                   ┘
IN      A,(C)       :ステータス読み出し
.
.
.
```

①現在のディスクヘッドのトラックNoが格納されているワークアドレスの内容をトラックレジスタに書き込む

0FF9H)。シークコマンドを，FDCのコマンドレジスタに格納する(I/Oポート0FF8H)。エラー時は5回リトライする。

③ 目的セクタ番号から(または目的セクタ番号へ)データをリード/ライトする。

リード動作時

目的セクタ番号を，FDCのセクタレジスタへ格納する(I/Oポート0FFAH)。

リードコマンドを，FDCのコマンドレジスタへ格納する(I/Oポート0FF8H)。

リストIV-2のように設定すると，FDCはディスク上の指定したトラック番号，セクタを検索して，目的の位置を見つけ出します。そして，DATA REQUEST信号(ステータスレジスタ1)のタイミングで，CPUに8ビット(1バイト)データを出力します。

CPUはこのデータをある一定時間(32μ秒)以内に取り込まなければなりません。1セクタ(256バイト)のデータを読み込むと，BUSY信号(ステータスレジスタ0)が“L”になり，リード動作が完了します。終了後はエラーチェックをしエラーの場合は5回リトライします。

ライト動作時

目的セクタ番号を，FDCのセクタレジスタへ格納する(I/Oポート0FFAH)。

リードコマンドを，FDCのコマンドレジスタへ格納する(I/Oポート0FF8H)。

上記のように設定すると，FDCはディスク上の指定したトラック番号，セクタ番号を検索して，目的の位置を見つけ出し，DATA REQUEST信号(ステータスレジスタ 1)のタイミングで，8ビットデータをFDCのデータレジスタへ格納します。8ビットデータはFDC内部でシリアルデータに変換されてディスクに書き込まれます。

DATA REQUEST信号から32μ秒以内にデータをデータレジスタに格納しないと，データ“00”が格納されたとFDCは判断しエラーとなります。1セクタ(256バイト)のデータを書き込むと，ライト動作が終了し，BUSY信号(ステータスレジスタ 0)が“L”でこれを判定します。終了後はエラーチェックをし，エラーの場合は5回リトライします。リード/ライトプログラムを作る際，1バイトデータのリード/ライトは，すべて32μ 秒でコントロールされているので，プログラムを32μ秒以内にディスクから(または ディスクへ)のリード/ライトを完了しなければなりません。

■ディスクリード/ライト プログラム

一般にディスク上の位置の指定は，前述のようにディスクサイド，トラック番号，セクタ番号を使って行います。しかしこの方法は便利とはいえません。任意の場所を指定するのに3つのパラメータを指定しなければならないからです。

そこでBASICなど一般のディスクオペレーティングシステムでは，ディスクの位置の指定に論理的レコードというものを使います。すなわちディスクサイド，トラック，セクタによって区分されたディスク上の各位置にレコード番号を論理的に割りつけるのです。

1レコードは256バイトを指し，物理的なセクタと容量は同じです。これによってディスクの任意の位置をレコード番号ひとつで指定できることになります。当然ディスクサイド，トラック番号，セクタ番号，レコード番号の間に一定の関係をもたせます。

X1 BASICの場合のこの関係は図IV-20のとおりです。

1レコードは1セクタと同じく256バイトですが，図IV-20からもわかるように，レコード番号は0から数えるのに対しセクタ番号は1から数えます。

レコード番号とディスクの物理アドレス量との間には，次のような関係があります。

レコード番号＝ディスクサイド＊16＋トラック番号＊32＋セクタ番号－1

rec＝16 ＊ Side＋32 ＊ track＋Secter－1

図IV-20 レコード番号と物理アドレスの相関

レコード番号	物理アドレス			備考
	ディスクサイド	トラック	セクタ	
0	0	0	1	トラック0，表
1	0	0	2	
2	0	0	3	
⋮	⋮	⋮	⋮	
15	0	0	16	
16	1	0	1	トラック0，裏
17	1	0	2	
⋮	⋮	⋮	⋮	
31	1	0	16	
32	0	1	1	トラック1，表
⋮	⋮	⋮	⋮	
47	0	1	16	
48	1	1	1	
49	1	1	2	
⋮	⋮	⋮	⋮	
1264	1	39	1	トラック39，裏
⋮	⋮	⋮	⋮	
1277	1	39	14	
1278	1	39	15	
1279	1	39	16	

レコード番号＝ディスクサイド＊16＋トラック＊32＋セクター1

この関係を使うと，レコード番号から次のようにディスクサイド，トラック番号，セクタ番号が求められます。

Secter＝(rec mod 16)＋1

track＝(rec ¥32)

Side ＝ {[rec－(rec mod 16)] mod32} ¥16

ディスクに関係するプログラムを作るときに，レコード番号からトラック番号，セクタ番号を，これらの関係にしたがって導き出すサブルーチンを作っておけば，ディスクの位置指定はレコード番号ひとつでできるようになります。

それではディスクリード，ディスクライトプログラム(リストIV-3)を紹介しましょう。

リストIV-3

```
DSERJP: DS      2
SPBUFF: DS      2
DSKTRK: DS      4
UNITNO: DS      1

;
;       DISK READ ROUTINE
;
;       DE = RECORD NO.
;       HL = BUFFER START ADDRESS
;       A  = NO. OF RECORD TO READ
;
;       NAME:DSKRED
;

DSKRED: PUSH    HL
        LD      HL,DSKERR
        LD      (DSERJP),HL
        LD      (SPBUFF),SP
;
        POP     HL
        CALL    RECNOC
        CALL    READY
AGNRDY: CALL    SEEK
REDAGN: LD      A,05H
REDRTY: PUSH    AF
        PUSH    HL
        LD      A,80H
        CALL    SETDCM
        PUSH    DE
        LD      DE,0F8FBH
        LD      C,E
        IN      A,(C)
        LD      C,D
MPNYMI: IN      A,(C)
        RRCA
        JR      NC,ENDRED
        RRCA
        JR      NC,MPNYMI
        LD      C,E
        IN      A,(C)
        LD      (HL),A
        INC     HL
        LD      C,D
        JR      MPNYMI
;
ENDRED: AND     04EH            :①
        POP     DE
        JR      Z,CNTRED
        POP     HL
        POP     AF
        DEC     A
        JP      Z,DIOER
        CALL    DRESET
        JR      REDRTY
;
CNTRED: POP     AF
        POP     AF
        EX      AF,AF'
        DEC     A
        JP      Z,MOTOFF
        EX      AF,AF'
        CALL    NXTSCT
        JR      NC,REDAGN
        JR      AGNRDY
;
;       DISK WRITE ROUTINE
;
;       DE = RECORD NO.
;       HL = WRTIE DATA START ADDRESS
;       A  = NO. OF RECORD TO WRITE
;
;       NAME:DSKWRT
;
DSKWRT: PUSH    HL
        LD      HL,DSKERR
        LD      (DSERJP),HL
        LD      (SPBUFF),SP
;
        POP     HL
        CALL    RECNOC
        CALL    READY
AGNWRT: CALL    SEEK
WRTAGN: LD      A,05H
WRTRTY: PUSH    AF
        PUSH    HL
        LD      A,0A0H
        CALL    SETDCM
        PUSH    DE
        LD      DE,0F8FBH
MPNWRT: IN      A,(C)
        RRCA
        JR      NC,ENDWRT
        RRCA
        JR      NC,MPNWRT
        LD      A,(HL)
        LD      C,E
        OUT     (C),A
        INC     HL
        LD      C,D
        JR      MPNWRT
;
ENDWRT: BIT     5,A
        JP      NZ,WRPTCT
        AND     7EH
        POP     DE
        JR      Z,CNTWRT
        POP     HL
        POP     AF
        DEC     A
        JR      NZ,WTRTYL
DIOER:  CALL    MOTOFF
        RET
;
```

```
WTRTYL: CALL    DRESET
        JR      WRTRTY
;
;
CNTWRT: POP     AF
        POP     AF
        EX      AF,AF'
        DEC     A
        JP      Z,MOTOFF
        EX      AF,AF'
        CALL    NXTSCT
        JR      NC,WRTAGN
        JR      AGNWRT



;
;
;       SUBROUTINES
;
;


RECNOC: PUSH    HL
        LD      L,A
        EX      AF,AF'
        LD      H,04H
        LD      A,0FFH
        DEC     L
        SUB     L
        LD      L,A
        OR      A
        SBC     HL,DE
        POP     HL
        JP      C,BADREC
        LD      A,E
RLLOP:  RLCA
        RL      D
        RLCA
        RL      D
        RLCA
        RL      D
        RLCA
        RL      D
        LD      A,E
        AND     0FH
        INC     A
        LD      E,A
        RET
;
READY:  PUSH    HL
        DI
        CALL    POWER
        LD      A,(UNITNO)
        AND     03H
        SRL     D
        JR.     NC,J1
        OR      10H
J1:     OR      80H
        LD      (UNITNO),A
        LD      C,0FCH
        OUT     (C),A
        PUSH    DE
DBSY:   LD      E,03H
        LD      HL,0000H
        LD      BC,0FF8H
RDYWT1: IN      A,(C)
        AND     81H
        JR      Z,RTRDYS
        DEC     HL
        LD      A,H
        OR      L
        JR      NZ,RDYWT1
        DEC     E
        JR      NZ,RDYWT1
        JP      DEVUNA
;
SEEK:   LD      C,0FBH
        OUT     (C),D
        PUSH    HL
        CALL    GETLST
        LD      A,(HL)
        LD      C,0F9H
        OUT     (C),A
        LD      (HL),D
        POP     HL
        LD      C,0F8H
        LD      A,1EH
        OUT     (C),A
DBUSY:  PUSH    HL
        PUSH    DE
        LD      B,20
SELF:   DJNZ    SELF
        LD      BC,0FF8H
BSYLP1: IN      A,(C)
        RRCA
        JR      C,BSYLP1
RTRDYS: POP     DE
        POP     HL
        RET
;
POWER:  PUSH    DE
        LD      D,0FFH
POWERL: LD      BC,0FFBH
        LD      A,0A5H
        OUT     (C),A
        LD      A,10
        CALL    DELAY0
        IN      A,(C)
        CP      0A5H
        JR      NZ,J2
        POP     DE
        RET
J2:     DEC     D
        JR      NZ,POWERL
        POP     DE
        JP      DEVUNA
;
DELAY0: DEC     A
        LD      HL,(1234H)
        JR      NZ,DELAY0
        RET
;
GETLST: LD      HL,DSKTRK
        PUSH    DE
        LD      A,(UNITNO)
        AND     0FH
        LD      E,A
        LD      D,00H
        ADD     HL,DE
        POP     DE
        RET
;
SETDCM: EI
        LD      C,0FAH
        DI
        OUT     (C),E
        LD      C,0F8H
        OUT     (C),A
        LD      A,07H
J3:     DEC     A
        JR      NZ,J3
        RET
;
MOTOFF: LD      BC,0FFCH
        LD      A,(UNITNO)
        AND     03H
        OUT     (C),A
        EI
        RET
;
DRESET: PUSH    AF
        PUSH    HL
        PUSH    DE
        CALL    GETLST
        LD      (HL),0
        XOR     A
        LD      BC,0FF9H
        OUT     (C),A
        DEC     C
        LD      A,02H
        OUT     (C),A
```

```
        CALL    DBUSY
        POP     DE
        CALL    SEEK
        POP     HL
        POP     AF
        RET
;
NXTSCT: INC     E
        LD      A,10H
        CP      E
        RET     NC
        LD      E,01H
        PUSH    HL
        LD      HL,UNITNO
        LD      A,(HL)
        XOR     10H
        LD      (HL),A
        AND     10H
        JR      NZ,BFIOST
        INC     D
        SCF
BFIOST: LD      C,0FCH
        LD      A,(HL)
        OUT     (C),A
        POP     HL
        RET
;
BADREC:
DEVUNA:
WRPTCT: CALL    MOTOFF
        LD      HL,(DSERJP)
        LD      SP,(SPBUFF)
        EX      (SP),HL
        RET
;
DSKERR: JP      0000H
;
;
END
```

:① DISK ERRORの処理。システムに応じて変更。ここではBOOT

ディスクリード・サブルーチン(DSKRED)

このサブルーチンは，Aレジスタに指定するレコード(セクタ)数のデータを，DEレジスタに指定するディスク上のレコード番号からHLレジスタに指定するメインメモリ上へ読み込みを行います。

このサブルーチンをコールする前に次の設定を行います。

DEレジスタ＝ディスク上のレコード番号

HLレジスタ＝ディスクから読み込むデータを格納するメインメモリの先頭アドレス

Aレジスタ　＝ディスクから読み込むレコード(セクタ)数

このサブルーチンはドライブ番号を指定するワークエリアとしてメモリの1バイトを使用し，UNITNOという名前をつけてあります。したがってサブルーチンコールの前にこのUNITNOワークエリアにあらかじめ，ドライブ番号(0…ドライブ0，1…ドライブ1)を入れておく必要があります。ディスク上のレコード番号16Hから8レコード分を，メインメモリのA000Hアドレスに読み込む例(リストIV-4)を示します。

リストIV-4

```
LD      DE,0016H
LD      HL,0A000H
LD      A,08H
CALL    DSKRED
.
.
.
```

ディスクライト・サブルーチン(DSKWRT)

このサブルーチンは，HLレジスタに指定するメインメモリのアドレスからのデータをDEレジスタに指定するディスク上のレコード番号の位置に，Aレジスタに指定するレコード数で書き込みます。このサブルーチンをコールする前に，次の設定を行います。

DEレジスタ＝ディスク上のレコード番号

HLレジスタ＝ディスクに書き込むデータのメインメモリの先頭アドレス

Aレジスタ　＝ディスクに書き込むレコード(セクタ)数

このサブルーチンも，コールする前にあらかじめUNITNOワークエリアにドライブ番

号を入れておかなければなりません。

メモリ上のA000HアドレスからA1FFHまでのデータをディスク上のレコード番号10Hに書き込む例(リストIV-5)を示します。

```
リストIV-5
LD      DE,0010H
LD      HL,0A000H
LD      A,02H
CALL    DSKWRT
.
.
.
```

A000H～A1FFHは512バイト。256バイトが1セクタですから，Aレジスタに02H(2セクタ)を指定するのです。

■X1 DISK BASICのディスク管理

ディスクの構成

X1のディスクは全部で1280レコード(0～1279)あります。DISK BASICでは，これらのレコードのうちいくつかをシステムで使用しています。

これらの領域とその役割りを説明します。図IV-21はその構成を示したものです。

レコード番号0のシステムインフォメーションには，IPL起動時にディスクからメインメモリにシステムプログラムを読み込むのに必要な情報が格納されています。

レコード番号1～13には，DISK BASIC本体が格納されています。

レコード番号32～1279までは，ユーザーファイルが格納されるエリアです。

FAT(File Allocation Table)

レコード番号14はFATと呼び，ディスク内のファイル管理(ファイルの格納位置の読み出し書き込み)に使用します。

X1では，ディスクアクセスの最小単位はレコードですが,DISK BASICでファイルを管理する場合は16レコード(256×16=4Kバイト)を最小単位として扱います。これをクラスタと呼びます。ファイルはクラスタ単位でディスクに書き込まれます。

X1のディスクは1280レコードあるので，クラスタ数に換算すると80クラスタということになります。

FATはディスク内の各クラスタの使用状況を示し，各ファイルがディスク上のどの位置のクラスタを，またどれだけの数のクラスタを使用しているかを示します。

図IV-21 DISK BASICディスクの構成

レコード番号	内容	備考
0	システムインフォメーション	IPL起動時に必要な情報を格納
1 ⋮ 13	DISK BASIC 本体	——
14	FAT	ファイル管理テーブル
15	reserved	
16 ⋮ 31	ディレクトリ	32バイト単位でファイルに関する情報を格納
32 ⋮ 1279	データ・エリア	ユーザーファイルを格納

レコード14の256バイトのうちの先頭から80バイトまでが，ディスク上の各クラスタの位置に対応しています。その中の値によってクラスタがどういう状況で使用されているかを知ることができるのです。

値が01H～7FHまでの場合は，ファイルがそのクラスタ内で終わらず別のクラスタにつながっていることを示しており，値そのものが次に使用するクラスタ番号を示しています。

あるファイルがクラスタ4から格納されているとし，そのときのFATが図IV-22のようになっているとします。

図IV-22　レコード14(FAT)

バイト	1	2	3	4	5	6	7	8	9
	02	03	05	06	89	07	08	85	----
対応クラスタ	0	1	2	3	4	5	6	7	8

この場合のファイルは，まず04クラスタを使用して，そのクラスタ内で終わらず続きがクラスタ06に格納されていることを示しています。クラスタ06を見ると07と入っています。ここでもファイルが終わらず，続きがクラスタ07に格納されていることを示します。

クラスタ07を見てみると，同じようにファイルがクラスタ08に続いていることがわかります。

このようにFATはディスクの使用状況をクラスタ単位(4Kバイト)で示しているのです。

ファイルがそのクラスタ内で終了している場合は，80H～8FHの値がそのクラスタに格納されます。その値から7FHを引いた値が，クラスタ内の何レコード目でファイルが終了しているかを示します。

前記の例では，クラスタ08が85Hとなっているので，このクラスタ内でファイルが終わっていることになり，さらに85H－7FH＝6ですから，クラスタ08の6レコード目でファイルが終わっていることを示します。

このファイルは，04H,06H,07Hの3つのクラスタのすべてと，クラスタ08Hの6レコードを使用していることがわかります。これをバイトになおすと，

$(4096\times3)+(6\times256)=13.5$Kバイト

の大きさのファイルということになります。

ディレクトリ(Directory)

レコード番号16から31までをディレクトリ領域と呼び，ディスクに格納されているファイルの目次の働きをします。

この領域に各ファイルのモード，ファイル名，ロードアドレス，作成年月日時分，ファイルが格納されている先頭クラスタ番号などの情報が格納されています。これによってファイル名とファイルの格納場所などの情報の対応ができるのです。

ひとつのファイルに対し32バイトが用いられています。その構成は1DH～1FHバイトを除いて，IPL用のインフォメーションブロックとまったく同じです。

1DH～1FHバイトはディスク上のファイルの格納されている先頭クラスタ番号が入ります。しかし実際に使われているのは1EHだけで，1DHと1FHバイトは常に00です。

このディレクトリ領域内には128個のファイルのディレクトリを書き込むことができま

す。個々のファイル情報は，図IV-23のとおりです。

ディスクの管理

DISK BASICでは，FATとディレクトリによってファイルを管理しているので，ファイルをセーブ/ロードするとき，DISK BASIC内で，これらの領域の参照もしくは更新がなされます。詳しく説明しましょう。

ロードする場合

ファイルをディスクからメモリへロードする場合，まずディレクトリ領域(レコード番号16～31)の中から，希望のファイルがあるかどうか調べなければなりません。

レコード番号16から1レコードずつディレクトリをメモリに読み込み，ファイルを探します。ファイル情報としてひとつのファイル当たり32バイト使用されているので1レコードには最大8つのファイル情報が入っています。

その8つのファイル情報のなかに希望するファイル名と一致するものがなければ，次のレコード(レコード番号17)を読み込みファイル名を探します。

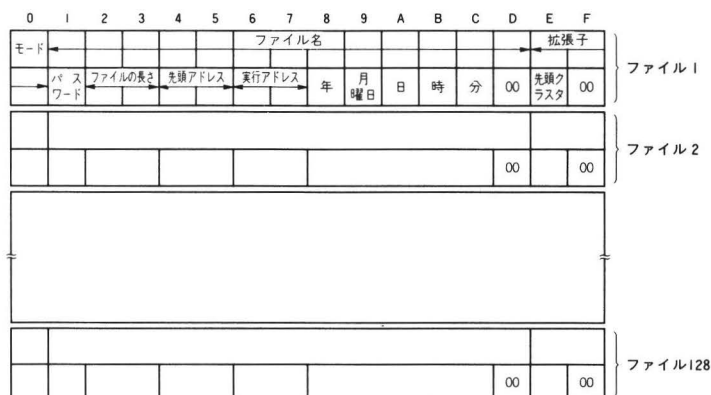
このようにして31レコード目までに希望のファイル名がない場合は，そのディスクには希望ファイルがないということになります。

もしファイル名が一致したら，ファイル情報の32バイトのうち31バイト目を参照してファイルの格納されている先頭クラスタ番号をとり出します。

レコード番号14のFATとファイルの情報からとり出した先頭クラスタ番号から，ファイルが格納されている状況がわかるので，ファイルが格納されているクラスタから順次データを読み出します。

各クラスタ内でファイルが終了しているかどうかを見ます。もしクラスタ内で終了していれば，そのクラスタに対応するFATのバイトの内容が80H～8FHの値になっているので，その値から7FHを引いた値のレコード数を読み込み，ファイルの読み込みを終了します。

ファイルがそのクラスタ内で終わっていなければ，4Kバイト分(16レコード)のデータ

図IV-23　ディレクトリ構成

を読み込み次のFATの内容を調べ，終了するまで読み込みをします。

読み込んだデータのメモリ上へのロードアドレスは32バイトのファイル情報のバイト20, 21にL，Hの順に入っています(たとえばバイト20, 21が80H, A0Hとなっていれば，読み込む際にそのファイルはA080Hアドレスから読み込まれます)。

セーブする場合

ファイルをディスクにセーブする場合は，まずディレクトリ領域(レコード番号16～31)を参照し，書き込もうとするファイルと同じ名前のファイルがあるかどうかを調べます。

もし，同じファイル名のものがあれば，インフォメーションブロックの31バイト目の先頭クラスタの位置を見て，ここからつながるFATの部分をすべて00にします。つまりレコード番号14のFAT領域のうちこのファイルが使用していた部分をすべて00とするわけです。次にこのファイルのインフォメーションブロックの場所にあたらしいディレクトリを書き込みます。

同じファイル名のファイルを消去した場合は，当然空き領域ができます(消去といっても見かけ上の表現で，実際にそのファイルを消去するのではない。ファイルのFATとDIR部分を消去すると，ファイルが行方不明になりファイルのアクセスができなくなる)。

しかし同じファイル名のものがない場合は，ディスク上に空き領域があるかどうかわかりません。そこでディレクトリ領域にインフォメーションブロックを書き込む領域があるかどうかを調べます。空き領域がなければディスクに書き込むことはできません。

レコード番号14のFAT領域を参照して，データ領域(空いているクラスタ)があるかどうかを調べます。空き領域がなければそのディスクに書き込むことはできません

ディレクトリ領域，データ領域ともに，空き領域があればディスクにデータを書き込めるのですが，この場合ファイルを1クラスタ(4Kバイト＝16レコード)ごとに区切って書き込みます。

レコード番号14のFAT領域をみて，空いているクラスタを探します。このとき，FATが00Hのところが空いているクラスタ位置を示しています。

1クラスタ分のデータをそのクラスタに書き込みますが，データが1クラスタ以内か，以上で異なります。1クラスタ以内であればデータはそのクラスタ内で終了しているので，ファイルを終了させる処理をします。1クラスタ以上であれば，そのクラスタにデータを書き込み，続きのデータを書き込むための空きクラスタをFATから探します。

空きクラスタがあれば，いまデータを書き込んだクラスタ番号を示すFATに次に書き込むクラスタ位置の番号を書きます。どのクラスタにファイルが続いているかを記録するわけです。データが1クラスタより少なくなるまで，1クラスタ単位で上記の操作を繰り返し書き込みをします。

書き込むデータが1クラスタ以内になったときのファイルの終了処理は次のようです。

ファイルの残りのデータが何レコード(1レコード＝256バイト)に相当するかを調べます。そのレコード分の残りのデータをディスクへ書き込みます。

レコード番号14のFATのクラスタ位置に対応するバイトに7FH＋レコード数を計算して書き込みます。

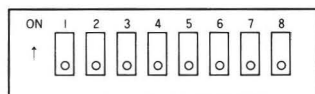
以上でファイルの書き込みが終了します。

2ドライブ以上の接続

CZ 800F には，フロッピディスクドライブが2基装備されています。X1 はインタフェース基板を通して一度に2ユニット(4ドライブ)までフロッピディスクドライブを接続することができます。

このとき各ドライブのプラント基板の上にある8回路DIPスイッチ(図IV-24, CZ-800F の上ブタをあけると上部に見える)を図IV-25の表のように設定する必要があります。

図IV-25は，4ドライブを接続するときのDIPスイッチの設定を示したものです。

図IV-24　フロッピディスク接続切り換えスイッチ

スイッチ番号	名　称	機　　能
1	TERM	最終ドライブのとき→ON，その他→OFF
2	DS0	DRIVE SELECT 0～3信号線を選択する（つまり，たとえばDS1をONにするとそのドライブはドライブ1として制御される）
3	DS1	
4	DS2	
5	DS3	
6	MX	ON：強制的にSELECTされる
7	HS	ON：DRIVE SELECT信号でR/Wヘッドロードが制御される
8	HM	ON：MOTOR ON信号でR/Wヘッドロードが制御される

HS，HMが共にOFFの時は，R/WヘッドロードはHEAD LOAD信号によって制御される

図IV-25　フロッピディスクドライブ
DIPスイッチ設定（4ドライブ接続時）

SW名称 ＼ ドライブナンバ	0	1	2	3
TERM	△	△	△	△
DS0	ON	×	×	×
DS1	×	ON	×	×
DS2	×	×	ON	×
DS3	×	×	×	ON
MX	×	×	×	×
HS	ON	ON	ON	ON
HM	×	×	×	×

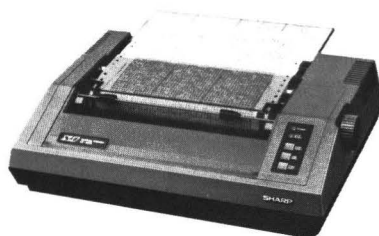
×：OFF
△：最終ドライブのみON，他は全てOFF

プリンタインタフェース

■X1のプリンタ仕様と動作手順

X1は，セントロニクス社製プリンタ仕様に準ずる8ビットパラレルインタフェースを標準装備しています。今では，ほとんどすべてのプリンタがセントロニクス仕様に準じています。

PhotoⅣ-9　ドットプリンタ CZ-800P

PhotoⅣ-10　漢字プリンタ CZ-80PK

この仕様は，制御信号などは一応統一されていますが，厳密な仕様規格があるわけではありません。実際細かいところでは少しずつ異なっています。

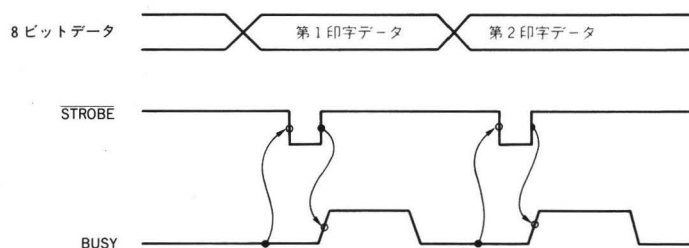
セントロニクスインタフェースは，D型36ピンという名称のコネクタ(DDK57シリーズ)を使用しています。

セントロニクスインタフェースは，8ビットパラレルデータを$\overline{\text{STROBE}}$信号とBUSY信号のふた組で制御するハンドシェーク方式を採用しています。図Ⅳ-26にセントロニクスのハンドシェークタイミング図を示します。

パソコン本体からプリンタへデータを出力するときは，おおよそ次の手順で行います。

① BUSY信号が“L”すなわちプリンタがデータを受けつける状態まで待ちます。

② BUSYが“L”ならば，8ビットデータを出力します。

図Ⅳ-26　セントロニクスのハンドシェークタイミング

③ $\overline{\mathrm{STROBE}}$信号を“L”にします。

④ $\overline{\mathrm{STROBE}}$信号を“H”にします。同信号の立ち上がりでプリンタはデータを受けつけます。このあとBUSY信号を“H”にして印字を行い，終了したらBUSY信号を“L”にすると，再びデータの受けつけができる状態になります。

⑤ さらにデータがあれば，再び①から始めます。

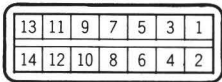
図IV-27,28はそれぞれ本体側および,CZ-800Pプリンタ側の接続端子を示したものです。

CZ-800Pは,X1シリーズ用プリンタとして発売されていますが,セントロニクスインタフェース仕様となっているので，これに準拠したコンピュータであればどれでも接続できます。

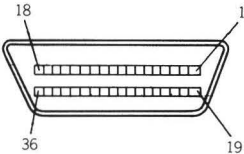
ペーパーエンド検出出力端子や初期設定入力端子などがあり，機能性の高いドットプリンタです。X1はセントロニクス信号のうち,$\overline{\mathrm{STROBE}}$, DATA0~7, BUSY, GND信号しか使っていません。

■プリンタ制御とコード

X1は,プリンタとのインタフェースをI/O空間の1A00H~1A02Hポートを使用して行います。プリンタインタフェースI/Oポートと制御信号は図IV-29のとおりです。

図IV-27 プリンタ接続端子と信号対応表
(本体側)

(本体裏側からみたときの配置)

ピン番号	信号名称	働き	本体よりみたときの信号方向
1	$\overline{\mathrm{STROBE}}$	プリンタにデータを読み込ませる開始信号	出力
2	DATA0(LSB)		出力
3	DATA1		出力
4	DATA2		出力
5	DATA3		出力
6	DATA4		出力
7	DATA5		出力
8	DATA6		出力
9	DATA7(MSB)		出力
10	NC	no connection	
11	BUSY	“L”:データを受けつけられる状態 “H”:データを受けつけられない状態	入力
12	NC	no connection	
13	GND		
14	GND		

図IV-28 プリンタ接続端子と信号対応表
(プリンタ側)

(本体裏側からみたときの配置)

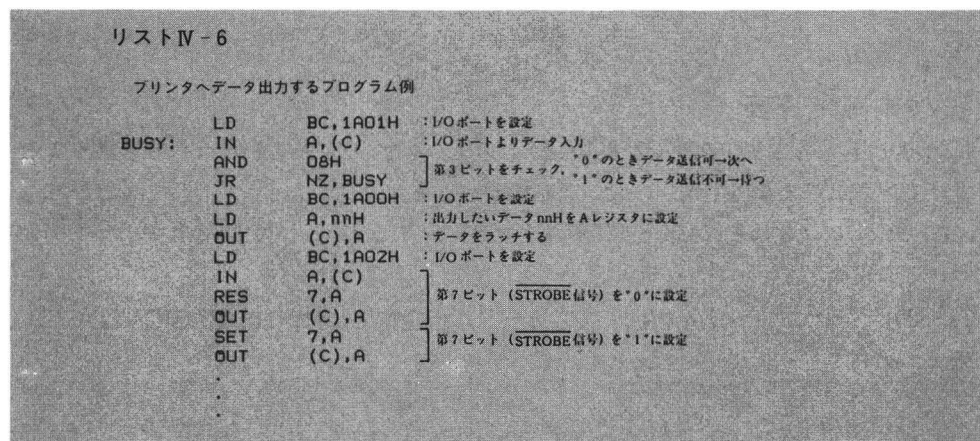
ピン番号	信号名称	プリンタからみたときの信号方向	ピン番号	信号名称	プリンタからみたときの信号方向
1	$\overline{\mathrm{STROBE}}$	入力	19	GND	—
2	DATA0(LSB)	入力	20	GND	—
3	DATA1	入力	21	GND	—
4	DATA2	入力	22	GND	—
5	DATA3	入力	23	GND	—
6	DATA4	入力	24	GND	—
7	DATA5	入力	25	GND	—
8	DATA6	入力	26	GND	—
9	DATA7(MSB)	入力	27	GND	—
10	$\overline{\mathrm{ACK}}$	出力	28	GND	—
11	BUSY	出力	29	GND	—
12	PAPER END	出力	30	GND	—
13	SELECT	出力	31	$\overline{\mathrm{INPUT \cdot PRIME}}$	入力
14	GND	—	32	$\overline{\mathrm{FAULT}}$	出力
15	NC	—	33	GND	—
16	GND	—	34	NC	—
17	CHASIS GND	—	35	NC	—
18	+5V	出力	36	NC	—

X1からプリンタへデータを出力する具体的なプログラム(リストIV-6)を示しておきます。

Hu-BASIC (X1/C/D) 上でのプリンタへの1バイト出力サブルーチンは，12DCH アドレスに対応しています。

リストIV-6

プリンタへデータ出力するプログラム例

```
        LD      BC,1A01H    :I/Oポートを設定
BUSY:   IN      A,(C)       :I/Oポートよりデータ入力
        AND     08H         ]第3ビットをチェック，"0"のときデータ送信可→次へ
        JR      NZ,BUSY     ]                      "1"のときデータ送信不可→待つ
        LD      BC,1A00H    :I/Oポートを設定
        LD      A,nnH       :出力したいデータnnHをAレジスタに設定
        OUT     (C),A       :データをラッチする
        LD      BC,1A02H    :I/Oポートを設定
        IN      A,(C)       ]
        RES     7,A         ]第7ビット（STROBE信号）を"0"に設定
        OUT     (C),A       ]
        SET     7,A         ]第7ビット（STROBE信号）を"1"に設定
        OUT     (C),A       ]
        .
        .
        .
```

図IV-29　プリンタインタフェースI/Oポートと制御信号

I/Oポート	信　号	対　応　ビ　ッ　ト
1A00H	出力データ8ビット	第7ビット DATA7 DATA6 DATA5 DATA4 DATA3 DATA2 DATA1 DATA0 第0ビット
1A01H	BUSY	第3ビット
1A02H	$\overline{\text{STROBE}}$	第7ビット

図IV-30　CZ-800P制御コード

分　類	記 号	コード(16進)	機　　能
印字命令	CR	0D	印字を開始して印字後改行しません
	LF	0A	印字を開始して印字後1行改行
	ESC, VT	1B, 0B, 30+n_1, 30+n_0	指定された行数だけダイレクトスキップを行います
フォームフィード	FF	0C	印字を開始して印字後改ページ
拡大	ESC, U	1B, 55	拡大(横2倍)文字を指定
キャンセル	CAN	18	バッファクリア
印字モード	ESC, R	1B, 52	10CPIピッチの印字を指定します
	ESC, E	1B, 45	12CPIピッチの印字を指定します
グラフィック	ESC, %, 2	1B, 25, 32, n_1, n_0	グラフィック印字開始指定 n_1, n_0：グラフィックデータ数（2バイト） n_1：上位バイト　n_0：下位バイト
改行幅	ESC, 6	1B, 36	6LPIピッチの改行指定
	ESC, 8	1B, 38	8LPIピッチの改行指定
	ESC, %, 9	1B, 25, 39, n	n/144インチの改行を指定
ページ長指定	ESC, F	1B, 46, 30+n_1, 30+n_0	½インチ数でのページ長設定
タブ	DC4	14	VFUにタブ位置をロードするための開始コード
	VT Channel No	0B, 31～3E	VFUにセットされたチャンネルNo.のタブ位置まで用紙を送ります
トップオブフォーム設定	ESC, 5	1B, 35	トップオブフォームを設定します
受信状態	DC1	11	装置をSELECT(受信可能)状態にします
	DC3	13	装置をDESELECT(ローカル)状態にします

このサブルーチンを使用するには，まず，Aレジスタに出力したいデータを設定し，このサブルーチンをコールします。

プリンタに41Hの1バイトデータを出力する例を次に示します。

```
LD      A,41H
CALL    12DCH
```

プリンタ制御コード表

CZ-800Pドットプリンタや CZ-80PK漢字プリンタは，X1から制御コードを送ることによって，さまざまな印字機能の設定，制御ができます。

図IV-30,31は，それぞれCZ-800Pドットプリンタおよび CZ-80PK漢字プリンタの制御コード一覧表です。

CZ-800Pドットプリンタを横拡大文字モードに指定するには，Hu-BASIC (X1/C/D)のプリンタ1バイト出力ルーチンを使って，リストIV-7のようにします。

```
リストIV-7
LD      A,1BH
CALL    12DCH
LD      A,55H
CALL    12DCH
.
.
.
```

CZ-80PK漢字プリンタをひらがな文字モードに指定するには，Hu-BASICのプリンタ1バイト出力ルーチンを使って，リストIV-8のようにします。

```
リストIV-8
LD      A,1BH
CALL    12DCH
LD      A,26H
CALL    12DCH
.
.
.
```

図IV-31　CZ-80PK制御コード

分類	記号	コード(16進)	機能
印字命令	CR	0D	印字のみ，または印字後改行
	LF	0A	印字後改行
	VT	0B	印字後改行
	ESC, VT	1B, 0B, 30+n_1, 30+n_0	印字後数行の改行 n_0, n_1：0～9
フォームフィード	FF	0C	改ページ
拡大	SO	0E	拡大(横2倍)指定
	ESC, U	1B, 55	拡大(横2倍)指定
	SI	0F	拡大解除

CZ-80PK制御コードつづき

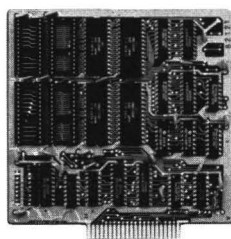
分　類	記号	コード(16進)	機　能
印字位置	HT	09	水平タブ
	POS	10, 30+n_2, 30+n_1, 30+n_0	キャラクタ単位での印字位置指定 n_0, n_1, n_2：0~9
	ESC, POS	1B, 10, 30+n_3, 30+n_2, 30+n_1, 30+n_0	ドット単位での印字位置指定 n_0, n_1, n_2, n_3：0~9
キャンセル	CAN	18	バッファクリア
印字モード	ESC, R	1B, 52	パイカ文字(標準文字)
	ESC, Q	1B, 51	縮小文字
	ESC, E	1B, 45	エリート文字
	ESC, H	1B, 48	コレスポンデンス文字
	ESC, P	1B, 50	コレスポンデンス文字
	ESC, K	1B, 4B, An, Bn	漢字　An：JISコード上位バイト Bn：JISコード下位バイト
グラフィック	ESC, %, 2	1B, 25, 32, n_1, n_0	8ビットドット列 (n n_0：データ数) n_1：上位バイト n_0：下位バイト
	ESC, I	1B, 49, 30+n_3, 30+n_2, 30+n_1, 30+n_0	16ビットドット列 n_0, n_1, n_2, n_3：0~9
外字の登録	ESC, *	1B, 2A, An_2, Bn, D1, D2, ……, D32	外字登録
キャラクタ・リピート	ESC, N	1B, 4E, 30+n_2, 30+n_1, 30+n_0, CH	同一文字の繰り返し n_0, n_1, n_2：0~9, CH：キャラクタコード
レフトマージン	ESC, L	1B, 4C, 30+n_2, 30+n_1, 30+n_0	レフトマージンの設定 n_0, n_1, n_2：0~9
改行幅	ESC, 6	1B, 36	1/6インチ改行モード
	ESC, 8	1B, 38	1/8インチ改行モード
	ESC, %, 9	1B, 25, 39, n	n/120インチ改行モード n：0~255
アンダーライン	ESC, X	1B, 58	アンダーライン指定
	ESC, Y	1B, 59	アンダーライン解除
グラフィックリピート	ESC, V	1B, 56, 30+n_3, 30+n_2, 30+n_1, 30+n_0, GD	8ビットドット列リピート n_0, n_1, n_2, n_3：0~9 n_3 n_2 n_1 n_0：リピート回数 GD：グラフィックデータ(8ビット)
	ESC, W	1B, 57, 30+n_3, 30+n_2, 30+n_1, 30+n_0, GD_1, GD_2	16ビットドット列リピート n_0, n_1, n_2, n_3：0~9 n_3 n_2 n_1 n_0：リピート回数 GD_1：グラフィックデータ(上側8ドット) GD_2：グラフィックデータ(下側8ドット)
ページ長指定	ESC, F	1B, 46, 30+n_1, 30+n_0	1/2インチ数でのページ長設定 n_0, n_1：0~9
水平タブ	ESC, (	1B, 28, ……	水平タブセット
	ESC,)	1B, 29, ……	水平タブ部分クリア
	ESC, 2	1B, 32	水平タブオールクリア
文字種類	ESC, $	1B, 24	カタカナモードの指定
	ESC, &	1B, 26	ひらがなモードの指定
文字間スペース	ESC	1B, n	ドットスペースの指定 n：1~6

漢字ROM

■漢字ROMボード

X1 turboには漢字ROMが内蔵されており，テキスト画面に表示できるようなハードウェアになっていますが，X1／C／Dは，漢字日本語表示のためのオプションとして，漢字のフォントデータを記憶した漢字ROMボード(CZ-8KR)を用意しています。このボードは拡張I/Oポートのスロットに装着して使用します。

PhotoⅣ-12
漢字ROM CZ-8KR

漢字フォントデータを得るには，まずJISコードや区点コードから，漢字フォントデータのROM内の収納アドレスを計算します。そしてI/Oポートを通して漢字ROMボードをアクセスし，漢字フォントデータを読み出してメモリに格納します。

ここで漢字JISコード，区点コードについてふれましょう。

漢字を数字コードで表すのに① JISコードによる方法，② 区点コードによる方法のふたつがあります。

JISコードは4桁の16進数で漢字と対応するのに対し，区点コードは4桁の10進数で漢字と対応します。たとえば“漢”という字をJISおよび区点コードで表すと3441(JISコード)と2033(区点コード)になります。

このJISコードと区点コードはそれぞれ独立しているのでなく，一定の相関を持っており一方から他方を導くことができます。

X1 Hu-BASICでは，KANJI$関数の引数として区点コードを採用しています。

JISコードを使用する場合，JISコードから漢字のフォントデータのROM内収納アドレスを算出することが簡単に行える利点があります。そのかわり漢字を4桁の16進数(したがって4～5桁の10進数)で表現しなければならず，扱いにくいところがあります。

JISコードに対して区点コードで漢字フォントデータの収納アドレスを算出するには，まず区点コードからJISコードを導き出し収納アドレスを算出します。すこし手間がかかるかわりに，ユーザーには表現しやすい(0～

図Ⅳ-32 漢字ROMアドレステーブル

JISコード上位バイト	最左列漢字フォンデータ収納アドレス	
	上位	下位
21	01	00
22	07	00
23	0D	00
⋮	⋮	⋮
28	2B	00
30	40	00
31	46	00
32	4C	00
33	52	00
⋮	⋮	⋮
4E	F4	00
4F	FA	00

9の4桁の数字)利点があります。

漢字ROMボードは「漢字コード表」上の非漢字および漢字のフォントデータのほか，JISコードの上位バイトと，漢字表上の行の最左列の漢字のフォントデータが収納されているROMアドレスに対応したアドレステーブルが書き込まれています。

図IV-32の表を使って，JISコードから漢字のフォントデータ収納アドレスを知ることができます。その計算を“字”を例に示します。

漢字フォントデータ・収納アドレスの計算

JISコードによる方法

① JISコード上位バイトから，30Hを引きます。ただしJISコード上位バイトが28H以下の場合は21Hを引きます。“字”のJISコードは3B7AHで上位バイトは3BHです。

3BH－30H＝0BH

② ①で得た値に0600Hをかけます。

0BH×0600H＝4200H

③ ②で得た値に4000Hを足します。ただしJISコード上位バイトが28H以下の場合は0100Hを足します。こうして得た値は，その行の最左列の漢字のフォントデータ収納アドレスになります。図IV-32はこのような値を示しています。

4200H＋4000H＝8200H

④ 求めたい漢字の収納アドレスの値を得るためには次の式によります。

収納アドレス＝(③から得た値)＋｛(JISコード下位バイト－20H)＊10H｝

“字”の場合，JISコード下位バイトは7AHなので，次のように示されます。

収納アドレス＝(8200H)＋｛(7AH－20H)＊10H｝(8200H)＋(5A0H)＝87A0H

BASICによるJISコード→収納アドレス計算プログラムをリストIV-9に示します。RUNすると漢字JISコードを聞いてきます。求めたい漢字JISコードを4桁の16進数で入力すれば，漢字ROM内の収納アドレスを計算して表示します。ループになっているのでやめたいときは，[SHIFT]＋[BREAK]を押します。

リストIV-10のプログラムは，マシン語プログラムによるJISコード→収納アドレス計算サブルーチンプログラムです。このプログラムを使うにはBCレジスタペアに漢字のコード2バイトを設定し，このサブルーチンをコールします。するとリターンしてくるときには，HLレジスタペアに収納アドレスが入っています。

```
LD      BC,****H          ;**** は漢字JISコード
CALL    JISCHG
```

HLレジスタの内容はIOCS(X1/C/D)の1202Hルーチンを使って表示することができます。

```
LD      BC,****H
CALL    JISCHG
CALL    1202H
```

リストⅣ-9　JISコードより収納アドレス算出プログラム(BASIC)

```
100 '
110 '  KANJI JIS CODE => KANJI ROM ADDRESS
120 '
130 DEFINT A-Z:CLS 4
140 PRINT"INPUT KANJI JIS CODE..&H";:Z$=INPUT$(4)
150 KCODE=VAL("&H"+Z$):IF KCODE=0 THEN GOTO 130 ELSE PRINT Z$
160 '
170 JISL=PEEK(VARPTR(KCODE))
180 JISH=PEEK(VARPTR(KCODE)+1)
190 IF JISL < &H20 OR JISL > &H7F THEN GOTO 140
200 IF JISH <= &H28 THEN ADD=&H100:DEC=&H21:GOTO 240
210 IF JISH < &H30 OR JISH > &H44 THEN GOTO 140
220 ADD=&H4000
230 DEC=&H30
240 '
250      JISH1=JISH-DEC
260      JISH2=JISH1*(&H600)
270      JISH3=JISH2+ADD
280 '
290      JISL1=JISL-&H20
300      JISL2=JISL1*(&H10)
310 '
320 ADDRESS=JISH3+JISL2
330 '
340 PRINT"-------------------"
350 PRINT"KANJI JIS CODE      : &H";HEX$(KCODE)
360 PRINT"KANJI ROM ADDRESS   : &H";HEX$(ADDRESS);"- &H0";HEX$(ADDRESS+&HF)
370 PRINT"-------------------"
380 GOTO 140
```

リストⅣ-10　JISコードより収納アドレス算出プログラム(マシン語)

```
;JISCHG ROUTINE
;SET KANJI JIS CODE IN BC REGISTER
;THEN CALL
;BC < 287FH OR BC > 3020H

JISCHG: LD      HL,3019H
        SBC     HL,BC
        JR      C,KCHR
        LD      HL,2880H
        SBC     HL,BC
        JR      C,EXIT
        LD      DE,2101H
        JR      KJAD
KCHR:   LD      DE,3040H
KJAD:   LD      A,B
        SUB     D
        ADD     A,A
        LD      B,A
        ADD     A,A
        ADD     A,B
        ADD     A,E
        LD      D,A
        LD      E,00H
        LD      A,C
        SUB     20H
        LD      H,00H
        LD      L,A
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,DE
EXIT:   RET
```

BC レジスタに設定されたJIS コードの漢字収納アドレスが計算されて表示されます。

このJIS コード→収納アドレス計算サブルーチンは，リロケータブルプログラムになっています。つまりどのアドレスからこのプログラムを書き込んでも，そのアドレスからコールすれば正しく実行するということです。

区点コードからJISコードへの変換

X1 ユーザーは，おそらくJIS コードより区点コードをよく使うことでしょう。しかし実際のプログラム内の操作は，区点コードよりJIS コードのほうがよく使われます。またHu-BASIC の場合でも，指定した区点コードは内部でJIS コードに変換されます。区点コードからJIS コードへの変換は次のように計算できます。

① JIS コードの上位バイト＝区点コードの区(16 進表示)＋20H

② JIS コードの下位バイト＝区点コードの点(16 進表示)＋20H

区点コードの区と点の意味は，区点コードの4桁のうち左2桁が区で，右2桁が点です。“字”の場合，区点コードは，2790で，

区＝27＝1BH → JIS コードの上位バイト＝1BH＋20H＝3BH
点＝90＝5AH JIS コードの下位バイト＝5AH＋20H＝7AH

となります。

したがって，結局“字”のJIS コードは3B7AH となります。

■漢字フォントデータの構成と読み出し

フォントデータの構成

漢字ROM ボードには，漢字のフォントデータが記憶されていますが，ある漢字のフォントデータを読み出すためには，フォントデータを収納しているROM 内のアドレスを求めなければなりません。

メインメモリの場合，ひとつのアドレスは1バイト(8ビット)を指すのに対して，漢字ROM ボードでは2バイト(16 ビット)を指します。これを1ワードといいます。

ひとつの漢字を構成するドットパターンは，この1ワードパターンを縦に16 個並べて作られています。そこで漢字を左右ふたつに分け1ワード(16 ビット)のうち，左側8ビットが漢字の左側部分を構成し，右側8ビットが漢字の右側部分を構成しています。

“阿”という字の場合，JIS コードは3024H なので，先の計算からすると漢字ROM ボード内の収納先頭アドレスは4040H です。

このアドレスは“阿”という字のフォントデータが入っている先頭アドレスを表しています。このアドレスから16 アドレスの範囲にフォントデータが入っていることを意味します。図IV-33 は，“阿”のフォントデータと漢字ROM 内アドレスの対応を示したものです。

フォントデータの読み出し

漢字フォントデータの漢字ROM 内収納アドレスの計算法がわかったところで，この計算法を使って漢字フォントデータを読み出す方法を説明します。

X1では，漢字フォントデータの読み出しに図IV-34のI/Oポートを使用します。

読み出し手順(図IV-35)は，次のとおりです。

図IV-33　漢字フォントデータの構成

① I/Oポート0E80Hを通して，収納アドレスの下位バイトを出力します。

② I/Oポート0E81Hを通して，収納アドレスの上位バイトを出力します。

③ I/Oポート0E82Hに01Hを出力し，漢字ROMボードをSELECT ON状態にします。

④ I/Oポート0E80Hを通して，左側バイトフォントデータを読み込みます。読み込んだデータをメモリバッファに入れます。

⑤ I/Oポート0E81Hを通して，右側バイトフォントデータを読み込みます。読み込んだデータを，先のメモリの次に入れます。この段階でハードウェアによって漢字ROMボードの内部アドレスがひとつカウントアップされます。これは連続的に漢字ROMボードからデータを読み出すときに，いちいちソフトウェアで漢字ROMボードの内部アドレスを加算しなくてすむように設計されているからです。

⑥ I/Oポート0E82Hに00Hを出力し，漢字ROMボードをSELECT OFF状態にします。

⑦ ③から⑥までの操作をあと15回繰り返し(合計16回)，フォントデータを読み込みメモリに入れていきます。前にメモリに入れたデータがなくならないように，読み出し動作(④，⑤)を行って，データをメモリに入れるたびに，メモリアドレス(に使用するレジスタの内容)をひとつ加算しなければなりません。

なお，②を実行したあと③を実行するのですが，この間3μS（マイクロセカンド＝10^{-6}秒)以上の時間間隔が必要です。これは漢字ROMボードをSELECT ON状態にするのに必要な時間と考えてもいいでしょう。

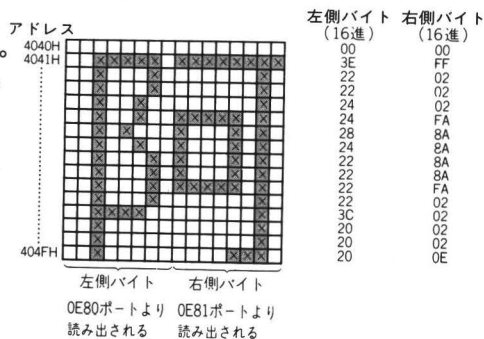
BASICとマシン語の漢字フォントデータ読み出しプログラムを2例(リストIV-11,12)紹介しましょう。

ひとつはBASICプログラムです。サブルーチンになっています。ADDRESS変数に漢字ROM内収納アドレスを設定して，

```
GOSUB 410
```

図IV-34　漢字ROMボードのアクセス法

I/Oポート	Z80CPUより入/出力	操　作　内　容
0E80H	OUT	収納アドレス下位バイト設定
	IN	漢字フォントデータ左側バイト読み出し
0E81H	OUT	収納アドレス上位バイト設定
	IN	漢字フォントデータ右側バイト読み出しと内部アドレスカウントアップ
0E82H	OUT	1．01H → 0E82Hに出力：ROMチップセレクトON 2．00H → 0E82Hに出力：ROMチップセレクトOFF, 増設用EPROMセレクト

を実行すれば，その漢字のフォントデータを読み出して，画面にイメージを表示するようになっています。

注意しなければならないのは，実行する前に変数Aを整数変数として定義することです。

```
400 DEFINT A:ADDRESS=&H87A0:GOSUB 410:END
```

と入力し，

```
RUN 400
```

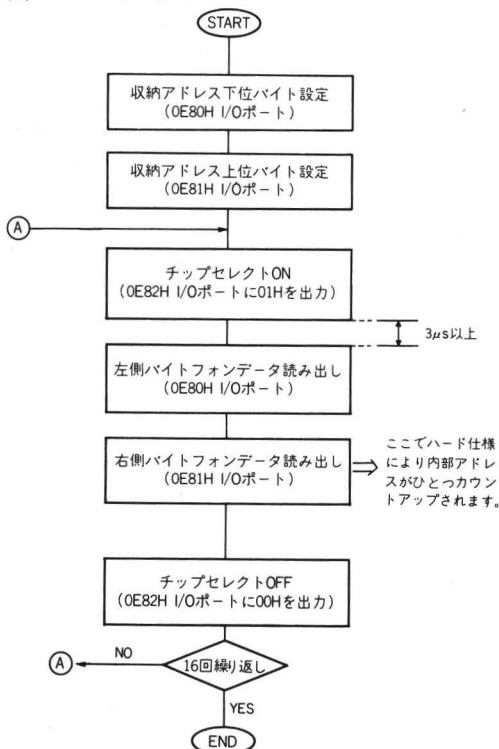
とすれば，画面に“字”のイメージが表示されます。&H87A0は“字”の漢字ROM内収納先頭アドレスです。

もうひとつはマシン語プログラムです。前のBASICと同じ動作をしますが，イメージ表示は行いません。このプログラムもサブルーチンになっており，これを使うにはDEレジスタに読み出しフォントデータを格納するメモリの先頭アドレスを，HLレジスタにフォントデータを求めたい漢字の漢字ROM内収納先頭アドレスを設定してコールします。

“阿”の字のフォントデータを求めてみます(図IV-36)。フォントデータを格納する先頭アドレスを

```
D000H
```

とします。まずROM内収納先頭アドレスを計算しなければなりませんが，ここでは，“阿”のJISコードを指定して，漢字ROM内収納先頭アドレスをJISCHGサブルーチン(リストIV-10)で求めます。次にリストIV-12のサブルーチンを使ってフォントデータを求めます。

図IV-35　漢字フォントデータフローチャート

図IV-36　読み出した漢字フォントデータ

漢字フォントデータ

```
MON
*DD000
:D000=00 00 3E FF 22 02 22 02
:D008=24 02 24 FA 28 8A 24 8A
:D010=22 8A 22 8A 22 FA 22 02
:D018=3C 02 20 02 20 02 20 0E
```

リストⅣ-11
漢字フォント読み出しルーチン

```
400 '
410 'KANJI FONT READOUT ROUTINE
420 '
430 'TO USE THIS ROUTINE , SET KANJI ROM ADDRESS IN "ADDRESS" VARIABLE
440 '
450 'THEN CALL.
460 '
470 '
480   ADDL=PEEK(VARPTR(ADDRESS))
490   ADDH=PEEK(VARPTR(ADDRESS)+1)
500   OUT &HE80,ADDL
510   OUT &HE81,ADDH
520   FOR LOOP=1 TO 16
530     OUT &HE82,&H1            'CHIP SELECT ON
540 '
550     DATL=INP (&HE80)
560     DATR=INP (&HE81)
570 '
580     OUT &HE82,&H0            'CHIP SELECT OFF
590    GOSUB "PRINT"
600   NEXT LOOP
620 RETURN
630 '
640 LABEL "PRINT"
650     DOT=DATL:GOSUB "PRINT IMAGE"
660     DOT=DATR:GOSUB "PRINT IMAGE"
670     PRINT
680 RETURN
690 '
700 LABEL "PRINT IMAGE"
710     FOR BIT=7 TO 0 STEP -1
720       IF DOT AND (2^BIT) THEN PRINT "*"; ELSE PRINT " ";
730     NEXT BIT
740 RETURN
```

リストⅣ-12
FONTRD サブルーチン（リロケータブル）

```
        TITLE   KANJI FONT READOUT
        ;       DE : FONT DATA BUFFER TOP ADDRESS
        ;       HL : KANJI ROM ADDRESS

FONTRD: LD      BC,0E80H    ]下位バイト設定
        OUT     (C),L       ]
        INC     BC          ]上位バイト設定
        OUT     (C),H       ]
        LD      H,L
KNJRD:  PUSH    HL
        LD      BC,0E82H    ]
        LD      A,01H       ]チップセレクトON
        OUT     (C),A       ]
        NOP                 :時間ディレー
        LD      BC,0E80H    ]
        IN      A,(C)       ]左バイト読み出し，メモリ収納
        LD      (DE),A      ]
        INC     DE          ]
        INC     BC          ]右バイト読み出し，メモリ収納
        IN      A,(C)       ]
        LD      (DE),A      ]
        INC     DE
        INC     BC
        XOR     A           ]
        OUT     (C),A       ]チップセレクトOFF
        POP     HL          ]
        INC     L           ]
        LD      A,H         ]
        ADD     A,10H       ]16回ループ
        CP      L           ]
        JR      NZ,KNJRD    ]
        RET                 ]
```

```
LD      BC,3024H     ："阿"の JIS コードを BC レジスタに設定
CALL    JISCHG       ：収納先頭アドレスを求める。結果はHLレジスタに入りリターン
LD      DE,0D000H    ：D000H アドレスからフォントデータ格納用に設定
CALL    FONTRD
```

このプログラムを実行すると，D000H アドレスから"阿"のフォントデータ 32 バイト分が格納されてリターンしてきます(図IV-36)。

Hu-BASIC でこのプログラムを実行した場合 MON コマンドの D コマンドで，メモリの D000H からダンプしてみてください。

BC レジスタに設定する JIS コードを，いろいろ変えて試してみてください。

漢字のフォントデータがわかれば何ができるでしょう。たとえばマシン語で漢字を表示することやドットプリンタに漢字を印字させたいときなどはフォントデータを使用します。ということは漢字に関することなら必ずフォントデータが必要だということです。

■増設用EP-ROMの読み出し

CZ-8KR 漢字 ROM ボードには，EP-ROM(8K バイト)を 2 個(ROM1,ROM2)基板上に増設できます。それらをアクセスするには，次の I/O ポートを使用します(図IV-37)。

図IV-37　増設用EP-ROMアクセス法

<table>
<tr><th>アドレス指定</th><th>上　位</th><th>下　位</th></tr>
<tr><td>(ROM1, ROM2同様)</td><td>I/Oポート0E81H
bit 7 6 5 4 3 2 1 0
× × × A12 A11 A10 A9 A8</td><td>I/Oポート0E80H
bit 7 6 5 4 3 2 1 0
A7 A6 A5 A4 A3 A2 A1 A0</td></tr>
<tr><td></td><td colspan="2">EPROM有効アドレス (8Kバイト = 2^13)
×：無効ビット</td></tr>
<tr><td>EP-ROMセレクト</td><td colspan="2">0E82H I/Oポートに00Hを出力
XOR A
LD BC, 0E82H
OUT (C), A</td></tr>
<tr><td>内部アドレスカウントアップ</td><td colspan="2">0E81H I/Oポートに対して入力コマンドを実行
LD BC, 0E81H
IN A,(C)</td></tr>
<tr><td>データ入力</td><td colspan="2">ROM1の場合：I/Oポート0E80Hを使用
ROM2の場合：I/Oポート0E81Hを使用</td></tr>
</table>

アドレスの指定から始めます。8K バイトなので 2^{13}=8K より，アドレスバスの A_0～A_{12} の 13 本で指定できます。下位の 8 本(A_0～A_7)は I/O ポート 0E80H から，上位 8 本(ただし有効なのは 5 本，A_8～A_{12})は I/O ポート 0E81H から出力してアドレス指定を行います。

アドレス 1234H を指定(図IV-38)したい場合は，次のようにします。

図IV-38　1234Hのアドレス指定

```
LD      BC,0E80H
OUT     (C),34H
INC     BC                    ;BC=0E81H
OUT     (C),12H
```

これはROM1, ROM2とも，まったく同じように指定できます。

次は増設用EP-ROMをSELECT ON状態にすることについてです。

これはI/Oポート0E82Hに00Hを出力することによって行います。このとき同時に漢字ROMはSELECT OFF状態になります。具体的には次のようにします。

```
XOR     A              ：Aを00Hにする
LD      BC,0E82H
OUT     (C),A
```

データの入力です。ROM1の場合はI/Oポート0E80Hを使用し，ROM2の場合はI/Oポート0E81Hを使用します。ただしROM2に対してデータ入力を実行した場合は，ハード仕様によって自動的に内部アドレスがカウントアップされます(ROM1は変わらない)。

```
ROM1:   LD      BC,0E80H
        IN      r,(C)          r：各レジスタ

ROM2:   LD      BC,0E81H
        IN      r,(C)
```

最後に増設EP-ROMのアクセスプログラム(リストIV-13)を紹介します。

リストIV-13

```
        LD      DE,RMDATA      ：データ格納メモリ先頭アドレス
        LD      HL,ADDRES      ：読み出しEPROMスタートアドレス
        LD      BC,0E80H     ┐
        OUT     (C),L        │
        INC     BC           │ 読み出しスタートアドレス指定
        OUT     (C),H        ┘
        INC     BC           ┐
        XOR     A            │ EPROM SELECT ON
        OUT     (C),A        ┘
        LD      H,L
RMRDLP: PUSH    HL             ：ROM1ならIOPORT＝0E80H
        LD      BC,IOPORT      ：ROM2ならIOPORT＝0E81H
        IN      A,(C)        ┐
        LD      (DE),A       ┘ 1バイト読み込み
        INC     BC           ┐ ROM1なら入力コマンド実行したとき，内部アド
        IN      A,(C)        │ レスがカウントアップされず，この部分必要
                             ┘ ROM2ならこの部分不要
        INC     DE             ：アドレスカウントアップ
        POP     HL
        INC     L
        LD      A,H
        ADD     A,nnH                 ;nnH＝読みたいバイト数
        CP      L
        JR      NZ,RMRDLP
  :
  :     (EXIT)
  :
```

このプログラムは，HLレジスタで指定するEP-ROM アドレスからデータを任意のバイト数(0～255)読み込み，DEレジスタで指定するメモリアドレスからデータを格納します。

ハドソン ソフトの日本語ワードプロセッサHuWPは，漢字ROMボード上の増設ソケットにEP-ROM2個を装着して使用するようになっています。

column 10

XOR A

XOR A を実行すると，どうしてAレジスタが0になるのでしょう。
XOR は，論理演算にかかわるひとつの記号で exclusive or (排他的論理和)といい，論理値表は 次のとおりです。

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

AとBが同じとき結果は0になり，AとBが違うときは1になることがわかります (排他的という言葉のゆえんです)。ここでAとBというのは XOR の入力の意味でAレジスタ，Bレジスタではないので念のため。
XOR Aは，Aレジスタ（8ビット）の各ビットと，Aレジスタの各ビットのXOR をとるというコマンドです。したがって各ビットは自分自身との XOR ですから，結果は0になるわけです。

より進んだシステムへ

X1の世界を広げる周辺機器群は豊富ですが，中でも忘れることのできないものにビデオ，ビデオディスクを有効に活用するためのビデオマルチプロセッサ(複数のビデオ機器のコントロールと高度なビデオ編集)，パーソナルテロッパ(高品質のスーパーインポーズ録画),デジタルテロッパ(スーパーインポーズ画像の録画とコントロール)があります。これらにより高度なビデオ編集が可能になります。

X1 turboの場合はデジタルテロッパの機能が内蔵されているのでスーパーインポーズを本体のみで行えるようになりました。

そのほかマウス，音響カプラ，ジョイスティックなどを接続することができます。

一方外部機器をX1でコントロールするためには，何らかのインタフェースカードを装着する必要があります。

このほか拡張I/Oボックス(CZ−81EB)が，本体内にI/Oスロットがない場合やより幅広い拡張のためにインタフェース基板を装着するためのものとして用意されています。

ここでは，マウスと並列・直列の両インタフェース機能を1枚のボードでもつシリアルパラレルインタフェースをとりあげそれぞれ説明します。

Photo V-13　デジタルテロッパ CZ-8DT

Photo Ⅳ-14　パーソナルテロッパ CZ-8DT2

Photo Ⅳ-15 マルチビデオプロセッサ CZ-8VPI

Photo Ⅳ-16　拡張I/Oボックス CZ-81EB

■マウスってなに？

最近注目されているデバイスに「マウス」があります。見かけは可愛らしい小さなボックス状のもので，手でおおうようにして押さえて動かします。

その動きがコンピュータに伝わり，ソフトウェアの力を借りて種々の働きをします。たとえばプログラムの中のメニューを選択するとき，マウスを使ってカーソルをメニュー内容の描かれている絵や文字のところに移動させることができます。お絵書きツールなどとして大変便利です。

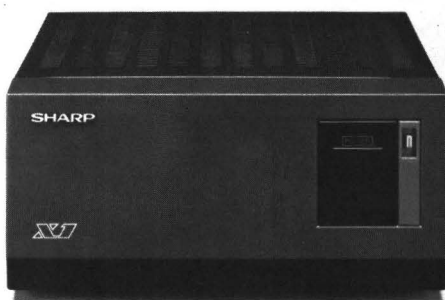
マウスの働きはハードよりむしろソフトによる部分が多いといえます。アップル社のスーパーパソコン「リサ」(製造中止)や「マッキントッシュ」の集約型ソフトにおけるマウス

の使いこなしはためいきが出るほどです。

Photo Ⅳ-17　マウス　MZ-1X10

外観図

内部回路

国産パソコンでは，シャープのMZ-5500(6500)で採用され次いでX1 turboにもインタフェースが装備されました。MZ-5500用のマウスは別売りされているので(MZ-1X10・図Ⅳ-39)，ぜひ自分のシステムにつないで試してみてはいかがでしょう。X1 turboの場合はマウスインタフェースが標準装備されていますのでマウスをそのまま接続することができます。

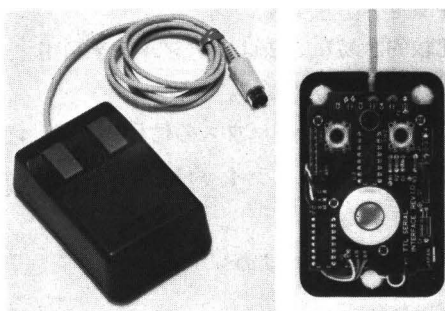
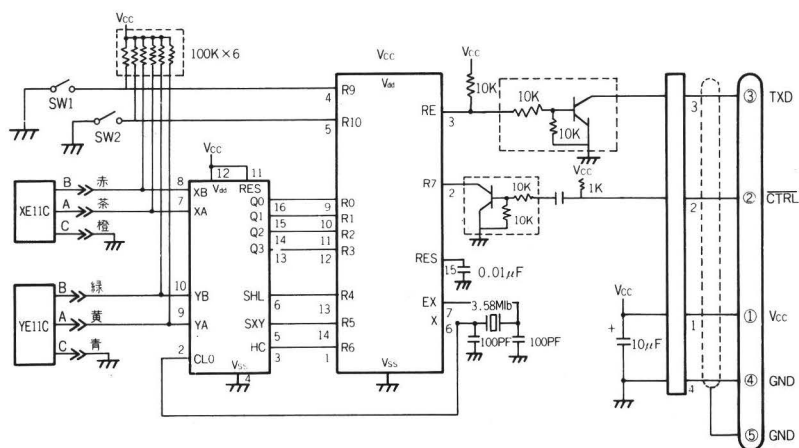
シャープのマウスは，本体内にパルスジェネレータ，カウンタ，そしてシリアルインタフェースが入っているのでデータはシリアルに伝送されます。

マウスのコントロールは次のようです。

マウスの位置の変位を知りたいときに，$\overline{\text{CTRL}}$信号をHIGHかLOWに落としてやります。すると前回ラッチした位置との変位およびスイッチのON,OFFがTXDから伝送されてきます。

そのフォーマットは図Ⅳ-40のとおりです。x方向，y方向とも，最上位ビットは正負を表しており，正負方向とも移動が80Hを超えるとキャリーフラグが立ちます(キャリーが0になる)。

ところでX1 turboは内蔵のシリアルインタフェースIC Z80A SIOを使うことで実施しています。

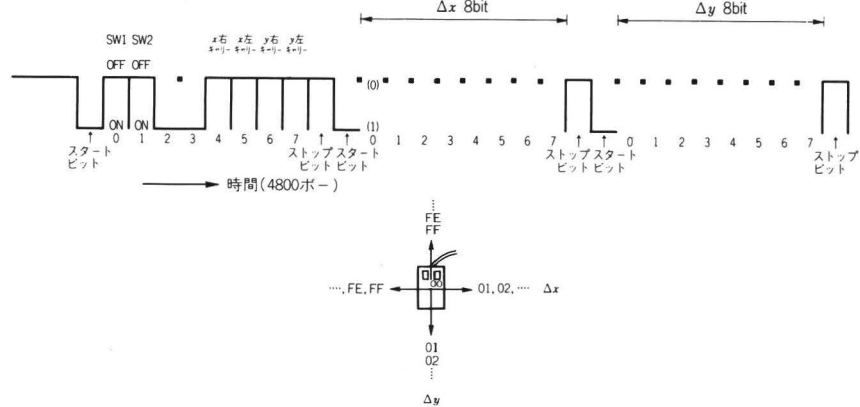
Z80A SIOはふたつの通信チャンネルを持ちそれぞれにデータポートとアドレスポート

図Ⅳ-39　マウス(MZ-1X10)内回路図

があります図IV-41に示すI/Oポートでコントロールします。

マウスをX1に接続する方法を考えてみます。

一番エレガントなのが，キーボード本体の伝送がシリアルなのを利用して，これにデータを乗せる方法です。

比較的容易な方法はオプションを利用する方法や自作のRS-232Cインタフェースボードを利用する方法です。

ただし，この場合マウスの信号がTTLレベルであるため，インタフェースボードの専用ラインレシーバ/ドライバ(1488,1489)を通す前のところに接続する必要があることに注意してください。

図Ⅳ-40　マウスのシリアル信号

図Ⅳ-4　X1turbo マウスI/0ポート

I/0ポート	内　容	
1F90	チャンネルA	データ
1F91		コントロール
1F92	チャンネルB	データ
1F93		コントロール

■シリアルパラレルインタフェース

外部機器を接続するときに大変役立つシリアルパラレルインタフェース(X1/C/D)を作ってみましょう。

シリアル伝送というのは，データを1本の導線で送る方式です。本体内部ではデータが8本の線で伝わるので，それを直列に直すためのICとして8251Aを使います。パラレルインタフェースのICとしては，有名な8255Aを用います。

シリアルパラレルインタフェース(図IV-42)はRS-232C規格(図IV-43)にします。いろいろなことができるようになりますが，たとえば次のようなことが考えられます。

① 音響カプラと電話機を用い，X1を大型コンピュータの端末として使う

② 直接あるいは①の方法でX1や他機種のコンピュータとつなぎ，プログラムファイルやメッセージの交換をする

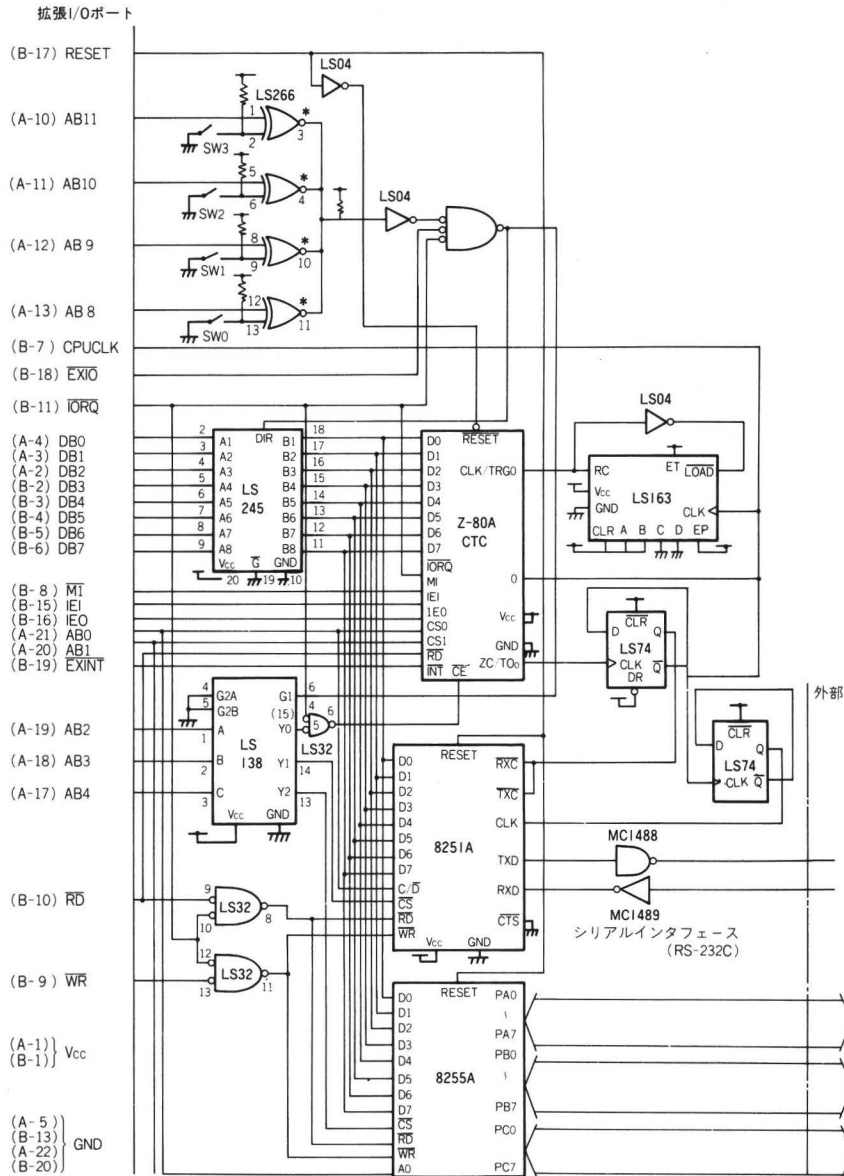
③　自作のコンピュータの端末として用いる

非同期のデータ転送の方式は図Ⅳ-44 のとおりです。

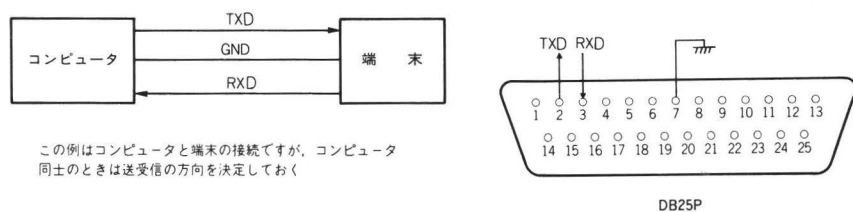
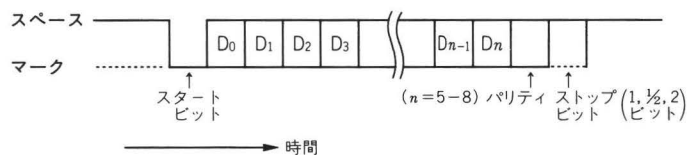
RS-232C はキッチリした規格となっていて，これが普及の原因となっているようです。

このインタフェースを作るためには，X1 専用のユニバーサルキットが必要です。カードエッジのピッチ幅が狭いので市販の KEL の大きなボードを切っても無理です。

なお RS-232C のボーレート(ビット/秒)を決める 8251A の $\overline{\text{RXC}}$, $\overline{\text{TXC}}$ へは Z80ACTC で作るクロックを入れています。

図Ⅳ-42　X1/C/Dシリアルパラレルインタフェース全回路図

図Ⅳ-43　RS-232Cコネクタおよび信号の役割

この例はコンピュータと端末の接続ですが，コンピュータ同士のときは送受信の方向を決定しておく

DB25P

図Ⅳ-44　非周期データ転送方式

V X1 IOCSルーチンを活用する

●

IOCSルーチンの活用
実際に利用してみよう
IOCSワークエリアの働きと操作法

IOCSルーチンの活用

■IOCSとは

X1のHu-BASICは，X1のハードウェアをサポートするため，特色のあるIOCSを備えています。BASICに内蔵されているモニタには強化されたエディタや文字列入力など，他機種のモニタには見られない多くの特徴を持っており，もっとも高性能なもののひとつとなっています。

この強力なHu-BASICモニタを手がかりに中核であるIOCSの働きとIOCS内ルーチンの使い方を見ていきます。またX1のマシン語操作法とIOCS内ルーチンを利用した簡単なマシン語プログラミングに挑戦してみましょう。

ところで，ひとつ注意しておきますとテープBASICでもディスクBASICでもいいのですが，カセットを装備していないと関係のないところがあります。またX1 turboはIOCSに相当する部分が本体ROM内にあり，アドレスが違っているため，対応するBIOS ROMのアドレスに変えないと本章のサンプルをそのまま実行することはほとんどできません。このため，X1 turboではX1用のテープ BASICを起動するか，後で述べるX1 MONITORを起動してください。

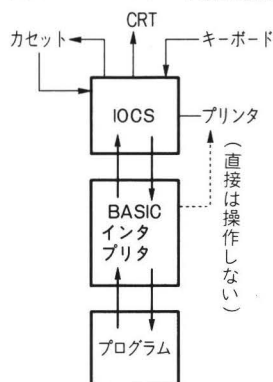
I/O空間をコントロールする方式はパソコンのハードウェアの違いによって大きく異なっています。たとえばキー入力処理，キー入力管理などをすべてサブCPUが行い，メインCPUは割り込みによってキー入力をするもの，CPUが直接キーボードを管理し，キーマトリクスコードからアルファベットのコード(ASCIIコード)への変換までメインCPUがすべて行うものなどいろいろあります。

このようにハードウェアの違いに合わせてI/Oを操作する基本ルーチンが作られますが，BASICのような独立したシステムソフトウェアでは，一般にI/O操作の基本ルーチンを1個所に集めてブロック化してあります。

この部分をIOCS(Input Output Control System)と呼びます。X1のHu-BASICでは，0000H～0FE1Hの間にまとめられています。

IOCSには，ディスク制御とグラフィックRAMの入出力などを除く多くの入出力ルーチンが含まれており，INPUT文，PRINT文などもIOCSを通じて実行されます(図V-1)。

もっともマシン語でI/O操作をする場合，これらをすべて自分で組んで行うこともできますが，その代わり常にハードウェアの

図V-1　IOCSの役割概念図

構造を意識しなければなりません。

それよりもI/Oを操作するルーチンはすべてIOCSを使うことにすれば，サブルーチンを呼ぶだけでハードウェアを気にせずにI/Oを操作することができます。そうなるとプログラムを組むのがとても楽になり，後でハードウェアの変更にも対処しやすいプログラムになります。

■なぜIOCSルーチンを使うか

IOCSルーチンをうまく使うとどのような効果があるかみてみましょう。

IOCSルーチンの中には入力パラメータの必要なものがあります。BASICでいえば，たとえば，

WIDTH　80

の『80』および，

INKEY(0)

の『0』が入力パラメータに相当します。

IOCSルーチンを利用するときには，パラメータはレジスタを介して渡します。つまりあるレジスタに適当な値をセットした後IOCSルーチンをコールするのです。

IOCSルーチンの中には，パラメータを持たないものもありますが，それはパラメータを設定しなくても仕事の内容が決まっている場合です。

ところでマシン語プログラムを作る場合，大きく分けてふたつの作り方があります。ひとつは，インタプリタが普通に行っている手順をそのままマシン語にする方法です。すなわち各レジスタに値をセットしてルーチンを次々とコールしていきます。この方法はマシン語を初めて始める人に比較的楽に扱える点が有利ですが，IOCSにすべて頼らなければなりません。

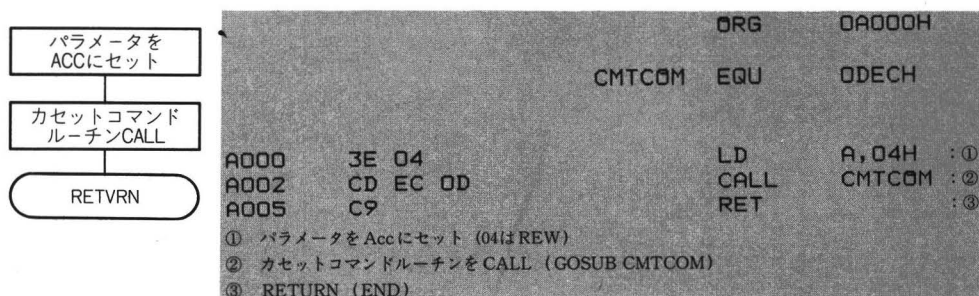
もうひとつは，I/O操作からすべてを直接マシン語で操作する方法です。ムダが少なく，スピードの速いマシン語プログラムが可能ですが，I/Oを操作するマシン語ルーチンをすべて自分で作る必要があります。

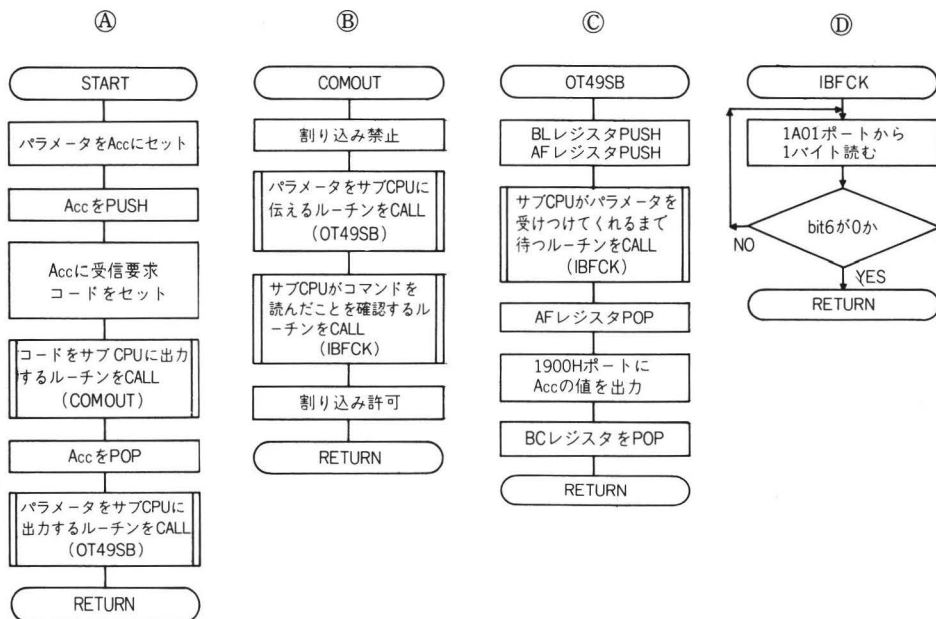
このふたつの方法の違いは，それぞれカセットの巻き戻しをするルーチンを組んで比べてみるとよくわかります。たとえばカセット内蔵タイプのX1についてみてみます。

まずIOCSルーチンを使う方法ですが，図V-2のようにわずか6バイトですみます。

図V-2　IOCSを利用する例・フローチャートおよびリスト（6バイト）

```
                              ORG    0A000H
                    CMTCOM    EQU    0DECH
A000    3E 04                 LD     A,04H    :①
A002    CD EC 0D              CALL   CMTCOM   :②
A005    C9                    RET             :③
```

① パラメータをAccにセット（04はREW）
② カセットコマンドルーチンをCALL（GOSUB CMTCOM）
③ RETURN（END）

次は，すべて自分で組む場合 START → COMOUT → OT49SB → IBFCK と何と 45 バイトものプログラムになってしまいます(図V-3)。

図V-3　自分ですべてプログラムする例・フローチャートとリスト(45バイト)

```
                              ORG     0A000H
   A000   3E 04                LD      A,04H
   A002   F5                   PUSH    AF
   A003   3E E9                LD      A,0E9H
Ⓐ A005   CD 0D A0             CALL    COMOUT
   A008   F1                   POP     AF
   A009   CD 16 A0             CALL    OT49SB
   A00C   C9                   RET

   A00D   FB          COMOUT:  DI
   A00E   CD 16 A0             CALL    OT49SB
Ⓑ A011   CD 23 A0             CALL    IBFCK
   A014   F3                   EI
   A015   C9                   RET

   A016   C5          OT49SB:  PUSH    BC
   A017   F5                   PUSH    AF
   A018   CD 23 A0             CALL    IBFCK
   A01B   F1                   POP     AF
Ⓒ A01C   01 00 19             LD      BC,1900H
   A01F   ED 79                OUT     (C),A
   A021   C1                   POP     BC
   A022   C9                   RET

   A023   01 01 1A    IBFCK:   LD      BC,1A01H
   A026   ED 78                IN      A,(C)
Ⓓ A028   E6 40                AND     40H
   A02A   20 F7                JR      NZ,IBFCK
   A02C   C9                   RET
```

結局，IOCS は I/O 操作に必要なルーチンのほとんどを含んでいるので，これを利用しない手はありません。IOCS 内ルーチンをうまく活用すると，強力なマシン語ルーチンでさえかなりコンパクトに組むことができ，マシン語が身近かなものになり活用範囲もグーンと広がります。

■IOCSを利用して楽々プログラミング

実際に IOCS 内のいくつかの主要なルーチンを用いて，小さなマシン語プログラムを作り，使い方を覚えてみましょう。

IOCS 内ルーチンを用いるマシン語プログラミングは，Z80CPU のレジスタにパラメータをセットし CALL するだけでよいので「マシン語プログラミングって思ったより簡単！」と思われることでしょう。

もっとも慣れないうちはハンドアセンブル(手と頭を使ってアセンブル言語からマシン語に変換すること)もけっこう大変な作業なので，はじめから大きなプログラムに取り組もうとせず，小さなプログラムをたくさん作ることから始めましょう。

ここではアセンブラを使っているわけではないので，メモリ効率を気にしすぎアクロバット的なプログラムを作るのは避けたいものです。少々メモリを多く使っても，あとで意味のわかるプログラムにするほうが大切です。ハンドアセンブルの場合は，とくにプログラムの中に 00H(NOP)を多く入れるなどして，あとで修正が楽なように工夫することが大切です。

IOCS 内のサブルーチンを CALL していく場合，各レジスタの値がサブルーチンから帰ってくるとき，保存されているとは限らないのでレジスタの保存状況には注意しなければなりません。

たとえばループカウンタにレジスタを使う場合など，保存したいのであれば PUSH 命令を用いて，レジスタをスタックというエリアに逃がしておいてからサブルーチンを CALL し，直後に POP 命令によって元に戻すなどの工夫が必要です。

レジスタの保存状況も含めて，この章の最後に IOCS ルーチンを X1/C/D/F と X1 turbo についてそれぞれ表にまとめてありますので参考にしてください。

■マシン語プログラムの入力の仕方

マシン語プログラムの入力は，Hu-BASIC の先頭に含まれているモニタを使うことにします。

Hu-BASIC を起動して MON ⏎ でモニタに入ります。

```
MON ⏎
*▨
```

このモニタの機能は強力で，簡単なマシン語開発ツールとしても役立ちます。モニタの持ついろいろな機能は，実際にマシン語ルーチンを入力しながらみていきます。

マシン語の入力の方法は，次のようなエディット書式でメモリの設定ができます(MあるいはDコマンドを使用します)。

```
:A000=3E 04 ⏎
アドレス データ データ ↑
                      リターンキー
```

ダンプリストやアセンブルリストの16進コードをたよりに打ち込んでください。

アセンブルリストとともに，ニモニックコードをBASIC的に表現して併記してあります。ただしBASICと完全に1対1対応ではありませんし，意味を表現しただけのもので参考的なものです。

マシン語のルーチンは，主にメモリのA000H番地から組んでいきますが，これはBASICの本体をこわさないためです。モニタは0FE2H～149FH番地の間にあり，マシン語だけを利用しようという人は，使っていない14A0H番地以降もすべてフリーエリアとして使うことができます。

モニタはBASICから分離して単独で使うこともでき，これだけでも立派なカセット・オペレーティング・システムを構成しているので，簡単なマシン語開発ツールとして利用できるわけです。

実際に利用してみよう

■画面出力

文字を出力する－1文字

ルーチン名　Acc PRT

アドレス　　0013H

画面に文字を表示してみます。

Acc(アキュームレータ)にセットされた値をASCIIコードとみなし，CRTに出力するルーチンです。BASICの1文字のPRINT文に相当します。

リストV-2を入力してみます。ここではメモリセット(M)コマンドを用います。Mコマンドも Dコマンドも，リストはエディット書式で出力するので，そのままスクリーンエディット(カーソルを修正したいところに持っていき変更後⏎キーを押す)でメモリの内容を設定することができます。

とくにMコマンドは，次のリストV-1のように2バイトでも3バイトでも空白を1個あけて，データを並べて書くことができます。

リストV-1

```
*MA000
:A000=3E FF ⏎
:A002=CD 13 00 ⏎
:A005=C9 ⏎
:A006=00 SHIFT + BREAK
*
```

入力部分

16進のデータとデータの間は見にくくなりますが，実はあけなくてもかまいません。これはHu-BASICのモニタと新しいシャープ系BASICのモニタの特有な機能で使いやすいところです。

リストV-2

```
                              ORG     0A000H
                      ACCPRT  EQU     0013H

A000    3E FF                 LD      A,0FFH          ; ACC=FFH
A002    CD 13 00              CALL    ACCPRT          ; GOSUB ACCPRT
A005    C9                    RET                     ; END
```

次にダンプ(D)コマンドでリストV-2を入力してみます。リストV-2は，リロケータブル(プログラムの先頭番地がどこでもよいという意味)なので，B000H番地から入力してみることにします。B000H番地からダンプして，次のようにスクリーンエディットしてください。

```
*DB000 B007 ⊘
:B000=3E FF CD 13 00 C9 00 00 ⊘ ／>↓へ..♪..
```
□＝入力部分

スラッシュ(／)の右側はスクリーンエディットの対象になりません。『／』の右側の文字は左側のマシン語をASCIIコードとみなして表示した文字列で文字列を探すときなどに利用できます。ダンプは開始番地だけを指定すると256バイト単位で行われますが，広い範囲をダンプする場合には，

```
*D14A0 9FFF ⊘
```
（14A0：開始番地　9FFF：終了番地）

のようにしてください。BREAKキーで一時停止します。

リストV-1を

```
*GA000 ⊘
```

と入力して実行すると，

```
〒
*
```

のように，CRTに〒が表示されます。A001HでAccにFFHをセットしているので，これに相当するキャラクタが表示されたわけです。

ここを30Hとすると，30HのASCIIコードに相当する0が表示されます。ASCIIコード表とにらめっこしながらA001Hをいろいろ変えて実行してみると，その様子がよくわかります。

注意しなければならないのは，Accにセットする値が00H～1FHのときは，文字が表示されるのではなく，そのASCIIコードに相当するコントロールコードが実行されることです。たとえば1CH～1FHまではカーソルコントロールとなっているので，カーソルの移動が実行されます。画面クリアやHOMEも実行できますし，07Hとするとベルが鳴ります。

画面制御はHu-BASICのモニタのユニークな機能ですから，うまく使うといろいろなことができます。ぜひ応用してください。

文字を出力する－文字列

ルーチン名　PRINT＃

アドレス　　000BH

指定したアドレスの文字列を出力します。

何文字にもわたる長いメッセージを出力する場合に，このルーチンを使います。

BASICでは，PRINT“メッセージ”，INPUT“メッセージ”や，あのSyntax error in などの“メッセージ”を出力するときに使われるルーチンです。

このルーチンを使うには，まず表示したいメッセージの文字列を，ASCIIコードでメモリ上のどこかに作る必要があります。ここではA040Hに作ってみました。DEレジスタにメッセージの先頭アドレスA040Hをセットしてコールすると，00Hがくるまで画面に出力します。

コントロールコードも実行されるので，文字列に入れてみましょう。

リストV-3を入力してください。

```
リストV-3
                                    ORG     0A000H
                           PRINT#   EQU     000BH
A000    11 40 A0                    LD      DE,0A040H        ;DE=A040H
A003    CD 0B 00                    CALL    PRINT#           ;GOSUB PRINT#
A006    C9                          RET

A040=0C D0 C5 BB DD BA DD C6 /.ミナサンコンニ
A048=C1 CA 0D DC C0 BC CA 20 /チハ.ワタシハ
A050=58 31 20 C3 DE BD 00 00 /X1 デス..
```

文字列を入力するときは，次のようなとても便利な機能がこのモニタに備わっています(リストV-4)。

```
リストV-4
*MA040 ⊘
:A040=0C;ミ;ナ;サ;ン;コ;ン;ニ;チ;ハ0D ⊘
:A04B=;ワ;タ;シ;ハ20;X;120;テ;゛;ス ⊘
:A056=00 SHIFT + BREAK
```

このように『；』(セミコロン)のあとに文字を書くと，ASCIIコードに変換されて入力できるので，長い文字列を作るときにはとても便利です。

文字列の最初の0CHは，コントロールコードで画面をクリアするものです。

```
*GA000
```

としてこのルーチンを実行すると，画面がクリアされ，

```
ミナサンコンニチハ
ワタシハ X1 デス
*▨
```

とメッセージが出力されます。

ここで，A04AHの0DHは改行として働きます。一般に改行は0DHと0AHが用いられますが，X1の場合は0AHを出力すると，カーソルから上がスクロールして改行します。これは今までBASICでスクリーンエディットをしているときに，不要な行接続が起きて困ることがありましたが，それを防ぐためのものです。

このルーチンが使えるようになると，画面に自在にメッセージが出力できるようになるので，マシン語プログラムらしくなってきます。

コントロールキャラクタの表示

ルーチン名　AccDIS

アドレス　04C8H

今までのプリントルーチンは，Accの値が00H～1FHまでのASCIIコードは，すべてコントロールコードとして処理されました。BASICのPRINT#0では，これらをコントロールコードとして実行せずに，コントロールキャラクタを表示しますが，このルーチンが

ちょうどそれに相当します。コントロールキャラクタや，矢印『→←↑↓』を表示したいときは，このルーチンを使います。

リストV-5は，256個のキャラクタをコントロールキャラクタを含めてすべて表示するルーチンです。

この中でA000HでXOR Aとしていますが，Aレジスタどうしでエクスクルーシブ OR(排他的論理和)をとるという意味です。フラグ変化を除けば，LD A,00H(3EH00Hの2バイト)と同じことですが，1バイトですむのでAレジスタを00Hにするときによく使われる方法です。

A00FHでコールしているサブルーチンCR1(04A7H)は，改行のルーチンです。改行ルーチンは，このほかにCR2(04A3H)があります。CR1は無条件で改行するのに対し，CR2はカーソル位置がディスプレイの左端にないときだけ改行します(リストV-5)。

リストV-5

```
                                ORG     0A000H

                        ACCDIS  EQU     04C8H
                        CR1     EQU     04A7H

A000    AF                      XOR     A               ;Acc=0
A001    47                      LD      B,A             ;B=Acc
A002    0E 10           L2:     LD      C,10H           ;C=10H
A004    CD C8 04        L3:     CALL    ACCDIS          ;GOSUB  ACCDIS
A007    3C                      INC     A               ;Acc=Acc+1
A008    05                      DEC     B               ;B=B-1
A009    28 0C                   JR      Z,EXIT          ;IF B=0 GOTO EXIT
A00B    0D                      DEC     C               ;C=C-1
A00C    20 07                   JR      NZ,L1           ;IF C<>0 GOTO L1
A00E    F5                      PUSH    AF              ;STAI=AF
A00F    CD A7 04                CALL    CR1             ;GOSUB  CR1
A012    F1                      POP     AF              ;AF=STAI
A013    18 ED                   JR      L2              ;GOTO   L2
A015    18 ED           L1:     JR      L3              ;GOTO   L3
A017    C9              EXIT:   RET
```

■キー入力

指定したアドレスに文字列を取り込む－1

ルーチン名　INPUTF

アドレス　　0003H

指定したアドレスにキーボードから文字列を取り込んでみます。BASICのLINEINP

リストV-6

```
                                ORG     0A000H

                        INPUTF  EQU     0003H
                        PRINT#  EQU     000BH
                        ACCPRT  EQU     1207H

A000    11 40 A0                LD      DE,0A040H       ;DE=A040H
A003    CD 03 00                CALL    INPUTF          ;GOSUB  INPUTF
A006    30 0B                   JR      NC,EXIT         ;IF Cy=0 GOTO EXIT
A008    F5                      PUSH    AF              ;STAI=AF
A009    11 14 A0                LD      DE,INPDAT       ;DE=INPDAT
A00C    CD 0B 00                CALL    PRINT#          ;GOSUB  PRINT#
A00F    F1                      POP     AF              ;AF=STAI
A010    CD 07 12                CALL    ACCPRT          ;GOSUB  ACCPRT
A013    C9              EXIT:   RET
A014    0D 41 63 63     INPDAT: DB      0DH,"Acc= ",00H
A018    3D 20 00
```

UT 文に相当するルーチンです（リストV-6)。

このルーチンは1行分のスクリーンエディットを行い，文字列をDEレジスタで示すアドレスに取り込みます。このときカーソルの前後に行が続いていると，そっくり全部を取り込んでしまうことは，LINEINPUT 文に似ています。A010H で 1207H をCALL していますが，これはAccの値を2バイトの16進数と見なして表示するルーチンです。

リストV-6を実行すると，カーソルが点滅して文字列の入力を要求してきます。文字列を入力してリターンキーを押すと，文字列を A040H 以降に取り込みます。

```
*DA040 A048
```

としてみると，その様子がよくわかります。

リストV-7

```
*GA000⊘
ABCDEF⊘
*▨
*DA040 A047⊘
:A040=41 42 43 44 45 46 00 00 /ABCDEF..
GA000
ASDF [SHIFT]+[BREAK]
Acc=03
*▨
```

注意しなければならないのは，

[SHIFT] + [BREAK]や，[CTRL] + [D]

による入力キャンセルの場合です。この場合には文字列は取り込まれません。つまりA040H 以降の値には変化がないのです。

この場合，INPUTF ルーチンから帰ってくるときにキャリーフラグが立ち，Accに，

[SHIFT] + [BREAK] ⟶ 03H

[CTRL] + [D] ⟶ 04H

がセットされます。

X1のIOCSの場合，IOCSルーチン実行中に何らかの異常があると，ルーチンから帰るときにキャリーフラグを立てて帰ってきます。異常の種類がAccにセットされているため，Acc値をみれば何の異常かがわかるようになっています。

指定したアドレスに文字列を取り込む－2

ルーチン名　BINPUT

アドレス　015AH

BASICのINPUT“T＝”のように，入力メッセージを必要とする場合，INPUTFでは不便な場合があります。BINPUT は，最大でもカーソルのある行以降の文字列しか取り込みません。入力メッセージと入力文字列を区別するため，DEレジスタが入力した文字列の先頭まで設定されて帰ってきます。INPUTF はメッセージを含めて1行分すべて取り込みますが，このとき行が続いていれば前の行まで含めて最大255文字までを取り込みます。

リストV-8を入力してください。

```
リストV-8
                                    ORG     0A000H

                           BINPUT   EQU     015AH
                           PRINT#   EQU     000BH
                           HLHXP    EQU     1202H

A000    11 18 A0                    LD      DE,MSSG1        ;DE=MSSG1
A003    CD 0B 00                    CALL    PRINT#          ;GOSUB  PRINT#
A006    11 40 A0                    LD      DE,0A040H       ;DE=A040H
A009    CD 5A 01                    CALL    BINPUT          ;GOSUB  BINPUT
A00C    D5                          PUSH    DE              ;STAI=DE
A00D    11 1F A0                    LD      DE,MSSG2        ;DE=MSSG2
A010    CD 0B 00                    CALL    PRINT#          ;GOSUB  PRINT#
A013    E1                          POP     HL              ;HL=STAI
A014    CD 02 12                    CALL    HLHXP           ;GOSUB  HLHXP
A017    C9                          RET
                                    ;
A018    44 41 54 41        MSSG1:   DB      "DATA= ",00H
A01C    3D 20 00
A01F    44 45 3D 26        MSSG2:   DB      "DE=&H",00H
A023    48 00
```

ここに出てくるHLHXPは，モニタ内ルーチンでHLレジスタの内容を4文字の16進数として表示するルーチンです。DEレジスタの内容を表示するために使っています。DEレジスタの値をHLレジスタに渡すため，A00CHでDEレジスタをPUSHし，A013HでHLレジスタにPOPしています。

このルーチンを実行すると，

```
DATA=␣▨
```

と入力を要求してきます。文字列を入力すると，DEレジスタの値が表示され，入力した文字列はA040H以降に取り込まれます。

```
:A040=44 41 54 41 3D 20 41 53 /DATA= AS
:A048=44 46 00 00 00 00 00 00 /DF......
```

こうしてみるとINPUTFとほとんど変わらないように見えます。ところが，大きく異なるのは，BINPUTルーチンから帰ってきたときのDEレジスタの値が，入力された文字列の先頭を示していることです。

上のリストの例では，DEレジスタの値はA046Hを示しているので，入力された文字列はASDFだということになります。

キーボードから1文字読み込む

ルーチン名　INKEY$

アドレス　　001BH

キーボードから文字を読み込んでみます。リアルタイムキースキャンや“スペースキーヲ　オシテクダサイ”といって，キー入力を待つ場合に使うルーチンです。BASICの“INKEY$”に相当します。

このルーチンをCALLするときにAccにパラメータをセットしますが，その内容によりHu-BASICの“INKEY$(　)”に相当する動作をします。

①AccにFFHをセット

このルーチンをCALLすると，それまでに押されたキーのASCIIコードをAccにセットして帰ってきます。何もキーが押されていないと00Hがセットされます。

X1の特徴である先行入力キースキャンなどで，Z80Aが他のルーチンの処理で忙しいときに使うルーチンです。入力されたキーを割り込み処理により先行入力バッファに蓄えておき，このルーチンをCALLしたときにバッファから読み出してAccにセットします。

②Accに00Hをセット

リアルタイムキースキャンをして，このルーチンをCALLしたとき，押されているキーのASCIIコードをAccにセットして帰ってきます。何もキーが押されていなければAccに00Hがセットされます。

③Accに01Hをセット

BASICではINKEY$(1)に相当します。

このルーチンをCALLしたとき，カーソルが点滅して1文字の入力があるまで待ち，入力があるとそのキーのASCIIコードがAccにセットされます。先行入力があった場合，その値を取り入れます。

④Accに02Hをセット

サブCPUから送られるファンクションコード部の，8ビットのデータがAccにセットされます。

BASICのINKEY$(2)と同じですが，リアルタイムキースキャンなので，CALLするときに押していないとFFHがセットされます。また[SHIFT]キー，[CTRL]キーなどのキー入力をコントロールするキーのみでデータキーが何も押されていないと，サブCPUからデータが送られてこないのでFFHがセットされます。

リストV-9のA004Hで，Accにパラメータをセットしています。この値を00H, 01H, 02Hと変えて各キー入力モードを試してみてください。

リストV-9

```
                           ORG     0A000H

                 INKEY$    EQU     001BH
                 ACCPRT    EQU     1207H

A000   CD 0C A0            CALL    DELAY          ;GOSUB  DELAY
A003   3E FF               LD      A,0FFH         ;Acc=FFH
A005   CD 1B 00            CALL    INKEY$         ;GOSUB  INKEY$
A008   CD 07 12            CALL    ACCPRT         ;GOSUB  ACCPRT
A00B   C9                  RET                    ;END
                           ;
A00C   01 00 00  DELAY:    LD      BC,0000H       ;BC=0000H
A00F   16 40               LD      D,40H          ;D=40H
A011   05        BLOOP:    DEC     B              ;B=B-1
A012   20 FD               JR      NZ,BLOOP       ;IF B<>0 GOTO BLOOP
A014   0D                  DEC     C              ;C=C-1
A015   20 FA               JR      NZ,BLOOP       ;IF C<>0 GOTO BLOOP
A017   15                  DEC     D              ;D=D-1
A018   20 F7               JR      NZ,BLOOP       ;IF D<>0 GOTO BLOOP
A01A   C9                  RET
```

■PCGとPSG

PCGをセットして表示する

ルーチン名　CGSET

アドレス　　002BH

プログラマブルキャラクタジェネレータ(PCG)は，X1の機能のうちでもっとも興味深いものです。

X1のPCGRAMは全部で6Kバイトもあります。通常256個のキャラクタをPCGによってセットする場合は，2Kバイトで十分なのです。ところが，X1はこれを6Kバイトとすることで，ドットごとに8色のカラーがセットできるようになりました。

従来のグラフィックRAMを用いたフルグラフィックスのほかに，X1ではまったく別の発想から生まれた，もうひとつのフルグラフィックスをテキスト画面に実現しています。

PCGのセットは，PCGのそれぞれのカラーに対してドットパターンを定義するため，フルカラーの場合，ひとつのキャラクタ定義には3回の書き込みが必要となります。

CGSETルーチンをコールするときは，各レジスタに次の値をセットします。

Dレジスタ　：セットしたい文字のキャラクタコード　00H～FFH

Eレジスタ　：アクセスするPCG RAMのI/Oポートアドレス上位15H～17H(14Hを指定すると，CGROMがアクセスされるが書き込みできない)

HLレジスタ　：ドットパターンデータバッファの先頭アドレス

CGSETルーチンから帰るときに，Dレジスタは保存されHLレジスタは＋8されるので，連続してパターンをセットする場合に便利です。

ドットパターンは，PCGをカラーでセットする場合，1色について8バイト，3色で計24バイトのデータが必要です。

ひとつのキャラクタは，8×8ドット(64ビット)で構成されています。したがって，これをデータにする場合，方眼紙に8×8のマス目を区切り，セットしたいパターンを作ったのち，上から順に1行ずつ8ドットごとに対応する2桁の16進数に換算していきます。

たとえば“◢”のキャラクタをセットする場合には図V-4のようにします。

したがってドットパターンデータ(16進)は，

“01, 03, 07, 0F, 1F, 3F, 7F, FF”

となります。

キャラクタを白で定義したい場合，同じパターンを青赤緑の各PCGRAMに書き込みます。カラーでセットする場合は3色作りますが，ドットの重なるところが混合色になるのは，BASICのDEFCHR$と同じです。

実際にキャラクタコード30Hと31Hにリンゴとさくらんぼの形を定義してみます。

図V-4　“◢”のビットパターンとデータ

MSB　　　　　　　　　　LSB

	7	6	5	4	3	2	1	0	データ
0								●	01
1							●	●	03
2						●	●	●	07
3					●	●	●	●	0F
4				●	●	●	●	●	1F
5			●	●	●	●	●	●	3F
6		●	●	●	●	●	●	●	7F
7	●	●	●	●	●	●	●	●	FF

リストV-10を実行すると，リンゴとさくらんぼがPCGに定義されたあと表示されます。ここではA040H以降の48バイトをドットパターンのデータバッファとし，A007H,A00CH,A011Hでそれぞれ青，赤，緑にセットしています。

PCGキャラクタの表示は，ワークエリア0026Hのカラーフラグに27Hをセットすることで行います。PRINTする直前にCGRAMモードにセットし，RETする直前に0026Hに07HをセットしてCGROMに戻しておきます。

さらに00H～1FHのキャラクタの表示は直接V-RAMとアトリビュートに書き込むか，AccDIS(04C8H)ルーチンを用いてください。

```
リストV-10
                                   ORG     0A000H
                                   ;
                          CGSET    EQU     002BH
                          ACCPRT   EQU     0013H

A000    21 40 A0                   LD      HL,0A040H      ;HL=A040H
A003    16 30                      LD      D,30H          ;D=30H
A005    1E 15             CGSLP:   LD      E,15H          ;E=15H
A007    CD 2B 00                   CALL    CGSET          ;GOSUB  CGSET
A00A    1E 16                      LD      E,16H          ;E=16H
A00C    CD 2B 00                   CALL    CGSET          ;GOSUB  CGSET
A00F    1E 17                      LD      E,17H          ;E=17H
A011    CD 2B 00                   CALL    CGSET          ;GOSUB  CGSET
A014    14                         INC     D              ;D=D+1
A015    7A                         LD      A,D            ;Acc=D
A016    FE 32                      CP      32H            ;IF A=32H THEN Z=1
A018    20 EB                      JR      NZ,CGSLP       ;IF Z<>1 GOTO CGSLP
A01A    3E 27                      LD      A,27H          ;Acc=27H
A01C    32 26 00                   LD      (0026H),A      ;COLOR=Acc
A01F    3E 30                      LD      A,30H          ;Acc=30H
A021    CD 13 00                   CALL    ACCPRT         ;GOSUB  ACCPRT
A024    3E 31                      LD      A,31H          ;Acc=31H
A026    CD 13 00                   CALL    ACCPRT         ;GOSUB  ACCPRT
A029    3E 07                      LD      A,07H          ;Acc=07H
A02B    32 26 00                   LD      (0026H),A      ;COLOR=Acc
A02E    C9                         RET
                                   ;
                                   ORG     0A040H

A040    00 00 00 00                DB      00H,00H,00H,00H,00H,00H,00H,00H
A044    00 00 00 00
A048    00 00 6E FF                DB      00H,00H,6EH,FFH,FFH,FFH,7EH,3CH
A04C    FF FF 7E 3C
A050    08 10 10 00                DB      08H,10H,10H,00H,40H,40H,20H,00H
A054    40 40 20 00
A058    00 00 00 00                DB      00H,00H,00H,00H,00H,00H,00H,00H
A05C    00 00 00 00
A060    00 00 00 00                DB      00H,00H,00H,00H,00H,00H,EEH,EEH
A064    00 00 EE EE
A068    18 0C 0A 0A                DB      18H,0CH,0AH,0AH,12H,24H,00H,00H
A06C    12 24 00 00
```

CGのドットパターンデータを読み出す

ルーチン名　CGREAD

アドレス　　0033H

CGROMやPCGRAMにセットされているドットパターンを読み出してみます。このデータは64ビットでできています。

このルーチンは，キャラクタコードをDレジスタに，読み出したいCGのI/OアドレスをEレジスタに，データバッファの先頭アドレスをHLレジスタにそれぞれセットしてコ

ールするとパターンデータを8バイトのデータとしてメインメモリのデータバッファに読み込んでくれます。またこのルーチンから帰るとき，Dレジスタは保存されHLレジスタは＋8されます。

実際にこのルーチンを使って，256個のPCGの全パターンを読み出すルーチンを組んでみましょう(リストV-11)。

リストV-11

```
                            ORG     0A000H
                   CGREAD   EQU     0033H
A000    21 00 B0            LD      HL,0B000H       ;HL=B000H
A003    11 15 00            LD      DE,0015H        ;DE=0015H
A006    D5         RDLOP0:  PUSH    DE              ;STACK=DE
A007    06 00               LD      B,00H           ;B=00H
A009    CD 33 00   RDLOP:   CALL    CGREAD          ;GOSUB  CGREAD
A00C    14                  INC     D               ;D=D+1
A00D    10 FA               DJNZ    RDLOP           ;IF B<>0 GOTO RDLOP
A00F    D1                  POP     DE              ;DE=STACK
A010    1C                  INC     E               ;E=E+1
A011    7B                  LD      A,E             ;A=E
A012    FE 17               CP      17H             ;IF A=17H THEN Z=1
A014    20 F0               JR      NZ,RDLOP0       ;IF Z<>1 GOTO RDLOP0
A016    C9                  RET
```

リストV-12

```
                            ORG     0A040H
                   CGSET    EQU     002BH
A040    21 00 B0            LD      HL,0B000H       ;HL=B000H
A043    11 15 00            LD      DE,0015H        ;DE=0015H
A046    D5         WRLOP0:  PUSH    DE              ;STACK=DE
A047    06 00               LD      B,00H           ;B=00H
A049    CD 2B 00   WRLOP:   CALL    CGSET           ;GOSUB  CGSET
A04C    14                  INC     D               ;D=D+1
A04D    10 FA               DJNZ    WRLOP           ;IF B<>0 GOTO WRLOP
A04F    D1                  POP     DE              ;DE=STACK
A050    1C                  INC     E               ;E=E+1
A051    7B                  LD      A,E             ;A=E
A052    FE 17               CP      17H             ;IF A=17H THEN Z=1
A054    20 F0               JR      NZ,WRLOP0       ;IF Z<>1 GOTO WRLOP0
A056    C9                  RET
```

こんどは，ドットパターンデータのバッファの先頭はB000Hにします。ここではPCGの青のデータを256個のキャラクタについて，全部読み出してから赤のデータを読み出すようにしています。したがってB000H～B7FFHが青，B800H～BFFFHが赤，C000H～C7FFHが緑のデータとなります。

BASICでDEFCHR$あるいはゲームを実行してPCGをセットしてください。このあとMONコマンドか，X1 MONITORをBOOTしてモニタに入ってください。このとき，CZ-800Cの場合，電源を切らない方法でBOOTする必要があります。CZ-801Cは電源を切ってもPCGRAMはしばらく保存されているようです。

リストV-11をA000Hから実行すると，PCGのデータがB000H～C7FFHに読み出されます。このデータをマシン語データとしてセーブすることもできますが，オートスタートアドレスは1000H(モニタスタート)か0000H(システムイニシャライズ)にしてください。

リストV-12は逆にB000H以降のドットパターンデータをPCGにセットするプログラムとなっています(リストV-12)。

このふたつのプログラムは，リストV-11のA009HのCALL CGREADをリストV-12のA049HでCALL CGSETに変えているだけです。またリストV-11でCGREADをコールするとき,Eレジスタにセットする値を14HとすることでCGROMを読むこともできます。

PSG出力停止

ルーチン名　PSGINT
アドレス　　013CH

ご存じのようにX1ではプログラマブルサウンドジェネレータ(PSG)が使われています。このPSGの制御は，PSGの各レジスタに必要なデータを設定することで行います。PSGにサウンドを出力させた後，停止をするのが通常ですが，このPSGINTルーチンはPSGの出力を停止させるものです。このルーチンでは,具体的にはPSGのレジスタ7〜10に次のようなデータを設定します。

レジスタ番号		データ
07H	←	37H(チャンネルA〜CのスイッチをOFFに)
08H	←	00H(チャンネルAの音量を0に)
09H	←	00H(チャンネルBの音量を0に)
0AH	←	00H(チャンネルCの音量を0に)

このルーチンは単にコールするだけで機能します。リストV-13は,すべてのレジスタを保存してPSGの停止を行う例です。

リストV-13

```
                              ORG     0A000H

                     PSGINT   EQU     013CH

A000    F5                    PUSH    AF              ;STACK1=AF
A001    C5                    PUSH    BC              ;STACK2=BC
A002    D5                    PUSH    DE              ;STACK3=DE
A003    CD 3C 01              CALL    PSGINT          ;GOSUB  PSGINT
A006    D1                    POP     DE              ;DE=STACK3
A007    C1                    POP     BC              ;BC=STACK2
A008    F1                    POP     AF              ;AF=STACK1
A009    C9                    RET
```

ベル音の発生

ルーチン名　BEEP
アドレス　　07F7H

「ピッ」というベル音を鳴らします。実際，これが[CTRL]+[G]およびBASICのBEEPコマンドの実行ルーチンです。このルーチンは単にコールするだけで機能します。

このルーチンでは07F8H番地からの2バイトの値を変更する(初期値は3000H)こと

で，ベル音の長さを変えることもできます。また，BCレジスタに任意の値を設定して，07FAH番地をコールすることでいろいろな長さのベル音を発生させることも可能です。

リストV-14はすべてのレジスタを保存してベル音を出力する例です。

```
リストV-14
                                ORG     0A000H

                        BEEP    EQU     07F7H

A000    F5                      PUSH    AF              ;STACK1=AF
A001    E5                      PUSH    HL              ;STACK2=HL
A002    C5                      PUSH    BC              ;STACK3=BC
A003    D5                      PUSH    DE              ;STACK4=DE
A004    CD F7 07                CALL    BEEP            ;GOSUB  BEEP
A007    D1                      POP     DE              ;DE=STACK4
A008    C1                      POP     BC              ;BC=STACK3
A009    E1                      POP     HL              ;HL=STACK2
A00A    F1                      POP     AF              ;AF=STACK1
A00B    C9                      RET
```

■カセットコントロール

カセットを制御する

ルーチン名　CMTCOM

アドレス　　0DECH

マシン語でカセットを操作してみます。

BASICでは“CMT＝”に相当するルーチンです。

このルーチンをコールする前にAccにセットするパラメータ00H～06H，0AHで，それぞれBASICのCMT＝Xのパラメータに対応しています。

BASICのCMTコマンドのパラメータと，IOCSのCMTCOMのパラメータの比較を示します(図V-5)。

図V-5　BASIC CMT/IOCS CMTCOMパラメータ比較表

BASIC	IOCS	働　き
CMTの値	CMTCOM Accセット値	
CMT＝0	00H	EJECT
CMT＝1	01H	STOP
CMT＝2	02H	PLAY
CMT＝3	03H	FAST
CMT＝4	04H	REW
CMT＝5	05H	APSS＋1
CMT＝6	06H	APSS－1
CMT＝10	0AH	REC，PLAY

リストV-15のサンプルを実行してみてください。

A001HのAccにセットするパラメータをいろいろ変えると，その様子がよくわかるはずです。0AHは実行すると，テープが消去されるので注意してください。

```
リストV-15
                              ORG     0A000H

                      CMTCOM  EQU     0DECH

A000    3E 00                 LD      A,00H           ;Acc=0
A002    CD EC 0D              CALL    CMTCOM          ;GOSUB  CMTCOM
A005    C9                    RET
```

ところでBASICからAPSSを実行すると実行中はキー入力を受けつけないことに気づいた人も多いでしょう。これはAPSSの実行中はサブCPUが忙しくてデータをZ80A CPUに送れないので，Z80AからサブCPUにデータを求める命令があると実行が止まってしまうからです。キー入力ルーチンをコールする場合などは気をつけてください。

APSSの実行中は，CMTCOMルーチンから戻ってこないので，これを利用してAPSSルーチンを組んでみましょう(リストV-16)。

```
リストV-16
                              ORG     0A000H

                      CMTCOM  EQU     0DECH
                      PRINT#  EQU     000BH
                      BINPUT  EQU     015AH
                      SPCTAC  EQU     1143H

A000    11 1C A0      START:  LD      DE,MSSG1        ;DE=MSSG1
A003    CD 0B 00              CALL    PRINT#          ;GOSUB  PRINT#
A006    11 40 A0              LD      DE,0A040H       ;DE=A040H
A009    CD 5A 01              CALL    BINPUT          ;GOSUB  BINPUT
A00C    CD 43 11              CALL    SPCTAC          ;GOSUB  SPCTAC
A00F    38 EF                 JR      C,START         ;IF Cy=1 GOTO START
A011    47                    LD      B,A             ;B=Acc
A012    C5            APSSLP: PUSH    BC              ;STAI=BC
A013    3E 05                 LD      A,05H           ;Acc=05H
A015    CD EC 0D              CALL    CMTCOM          ;GOSUB  CMTCOM
A018    C1                    POP     BC              ;BC=STAI
A019    10 F7                 DJNZ    APSSLP          ;B=B-1:IF B<>0 APSSLP
A01B    C9                    RET
                              ;
A01C    41 50 53 53   MSSG1:  DB      "APSS= ",00H
A020    3D 20 00
```

A009HでBINPUTを用いて，キーボードからループ回数を取り込みます。

DEレジスタの示す最初の文字は数字なので，これを16進数に変換しなければなりません。ここでモニタ内ルーチン1143Hを用います。これはDEで示すアドレスの文字を16進数に変換し，Accにセットしてくれるのです。

このルーチンで数字以外の文字を変換しようとすると，キャリーフラグが立つので，そのエラー処理も欠かせません。

A011HでBレジスタにループ回数をセットし，A01AHまでがループになっています。DJNZ命令は，Bレジスタのデクリメントとゼロの判断をして相対ジャンプをする便利な命令で，よく使われます。

カセットの状態を検出する

ルーチン名　CMTSNS

アドレス　　0DF6H

カセットをコントロールすることができたので，こんどはカセットの状態を検出してみましょう。

X1では，カセットの状態の検出はサブCPUが常に行い記憶しています。このルーチンはサブCPUと通信を行い，カセットの状態を受けてAccにセットして帰るルーチンです。

Accの各ビットは，次のようにカセットの状態に対応しているので，Accの内容をみればカセットの状態がわかります。BASICでは“CMT(　)”に相当しています(図V-6)。

図V-6　カセット状態データのビット構成

MSB　LSB

7	6	5	4	3	2	1	0
無意味					PROTECT	TAPE SET	MOTOR

Acc bit 0……0ならMOTOR OFF(テープエンド)。

Acc bit 1……0ならテープが入っていない。

Acc bit 2……0ならPROTECTのツメが折ってある。

リストV-17を実行すると，カセットが先頭まで巻き戻され消去が始まります。A00BHでカセットの状態を検出し，巻き戻しが終わるまでループを繰り返します。

Accの各ビットが1か0かの判断は，ビットテスト命令BITを用いています。

リストV-17

```
                          ORG     0A000H

                  CMTSNS  EQU     0DF6H
                  CMTCOM  EQU     0DECH
                  PRINT#  EQU     000BH

A000  3E 04               LD      A,04H                ;Acc=04H
A002  CD EC 0D            CALL    CMTCOM               ;GOSUB  CMTCOM
A005  11 25 A0            LD      DE,MSSG1             ;DE=MSSG1
A008  CD 0B 00            CALL    PRINT#               ;GOSUB  PRINT#
A00B  CD F6 0D    SNSLOP: CALL    CMTSNS               ;GOSUB  CMTSNS
A00E  CB 47               BIT     0,A                  ;IF A(0)=0 THEN Z=1
A010  20 F9               JR      NZ,SNSLOP            ;IF Z<>0 GOTO SNSLOP
A012  3E 0A               LD      A,0AH                ;Acc=0AH
A014  CD EC 0D            CALL    CMTCOM               ;GOSUB  CMTCOM
A017  CD F6 0D            CALL    CMTSNS               ;GOSUB  CMTSNS
A01A  CB 57               BIT     2,A                  ;IF A(2)=0 THEN Z=1
A01C  20 06               JR      NZ,EXIT              ;IF Z<>0 GOTO EXIT
A01E  11 36 A0            LD      DE,MSSG2             ;DE=MSSG2
A021  CD 0B 00            CALL    PRINT#               ;GOSUB  PRINT#
A024  C9          EXIT:   RET
                          ;
A025  52 45 57 49 MSSG1:  DB      "REWIND AND ERASE",0DH,00H
A029  4E 44 20 41
A02D  4E 4D 20 45
A031  52 53 45 0D
A035  00
A036  57 52 49 54 MSSG2:  DB      "WRITE PROTECT!",00H
A03A  45 20 50 52
A03E  4F 54 45 43
A042  54 21 00
```

この命令は，レジスタの各ビットをみて1か0かの判断をするものです。BIT 0，Aとすると，Aレジスタの第0ビットをみて，これが1ならゼロフラグが立つという命令です。

ファイルコントロールブロックのロード

ルーチン名　LOAD1

アドレス　0041H

ファイルコントロールブロック(FCB)は，プログラムをLOADやSAVEするとき，プログラムの最初に記録されている短い情報部分のことをいいます。

この部分は，別名ヘッダとも呼ばれ，32バイトの情報を含んでいます。ファイル属性，ファイルネーム，パスワード，プログラムサイズ，ロード開始アドレス，オートスタートアドレス，制作年月日および時刻の情報が書き込まれています。

このFCBを操作できるようになると，マシン語の応用範囲がグーンと広がります。たとえば，マシン語プログラムを任意のアドレスに作って，別のアドレスから実行させたりロードしてからつなぎ合わせる，といった高度な使い方も可能になってきますのでしっかり覚えましょう。

このLOAD1ルーチンは，BCレジスタにFCBの長さを，HLレジスタにロード開始アドレスを，それぞれセットしてコールするとHLレジスタで示すメインメモリのアドレスにFCBをロードしてくれます。Hu-BASICは，FCBの長さが20H(32バイト)，ロード開始アドレスがFF00H(ワークエリア)になっています。

ここではA040HにFCBロードしてみましょう。カセットレコーダにCZ-8CB01をセットしリストV-18を実行してください。

FCBのロードが終わると，レコーダがストップするので，DコマンドでA040H以降をのぞいてみてください。

FCBの構造はリストV-19のようになっています。この構造はX1ではテープだけでなく，フロッピディスクおよびEMMボードもすべて同じです。

リストV-18

```
                                 ORG     0A000H

                         LOAD1   EQU     0041H
                         CMTCOM  EQU     0DECH

A000     21 40 A0                LD      HL,0A040H       ;HL=A040H
A003     01 20 00                LD      BC,0020H        ;BC=0020H
A006     CD 41 00                CALL    LOAD1           ;GOSUB  LOAD1
A009     3E 01                   LD      A,01H           ;Acc=01H
A00B     CD EC 0D                CALL    CMTCOM          ;GOSUB  CMTCOM
A00E     C9                      RET


:A040=01 42 41 53 49 43 20 43 /.BASIC C
:A048=5A 38 43 42 30 31 20 20 /Z8CB01
:A050=20 20 10 9E 00 00 00 00 /   .点....
:A058=82 B3 17 23 50 00 00 00 /Σウ.#P...
```

① 最初の1バイト：ファイル属性で01は，マシン語ファイルを示します。

01；マシン語ファイル　　　(Bin)
02；BASICテキストファイル (Bas)
04；ASCIIファイル　　　(ASC)

② ファイルネーム13バイト：ASCIIコードで文字列が入ります。ファイルネームが13文字以下の場合，余りの部分は20Hが入ります。

③ 拡張ファイルネーム：ファイルネームで点に続く3文字がここに入ります。指定なしの場合は20Hが入ります。

④ パスワード：ファイルネームのあとの『；』(セミコロン)に続く文字がパスワードとみなされ暗号化されて入ります。パスワードなしの場合は20Hが入ります。

⑤ プログラムのサイズ：L,Hの順で書かれています。マシン語およびBASICテキストファイルのときだけ有効です。

⑥ ロード開始アドレス：L,Hの順で書かれています。マシン語のファイルのときだけ有効です。

⑦ オートスタートアドレス：L,Hの順で書かれています。マシン語のファイルのときだけ有効です。

⑧ 作成年月日，時刻：この例では年，日，時，分はBCD進で記録されます。月，曜日は16進で記録され，82年11月17日23時50分水曜日になります(図V-7)。

図V-7　作成年月日，時間データ

年	年	月	曜日	日	日	時	時	分	分
8	2	B	3	1	7	2	3	5	0

⑨ ファイル格納アドレス：テープの場合には意味を持ちませんが，フロッピディスクやEMMボードの場合に，ファイルが格納されている一番最初のレコード番号(File Allocation Table)を示し，H，L，Mの順で記録されています。テープでは，000000Hにしておいてください。

このLOAD1ルーチン実行中に何らかのエラーがあると，キャリーフラグがセットされます。エラーの内容はAccをみればわかるようになっています。

Accの内容の意味

Acc＝00H……エラーなし　OK!
　＝01H……BREAKされた
　＝02H……チェックサムエラー

＝03H……テープがセットされていない

＝04H……なし (WRITE, PROTECT SAVEのみ)

＝05H……テープ終了あるいはカセットキーが押された

このエラーに関する情報は，あとで紹介するLOAD2, LOAD2?, SAVE1, SAVE2でも同じ値がAccにセットされます。

データブロックのロード

ルーチン名　LOAD2

アドレス　　0044H

テープに記録されているプログラムを，オーディオレコーダで聞いてみると，FCBとデータブロックのふたつのブロックからできていることがわかります。

このルーチンは，そのうちのデータブロックをロードするルーチンです。FCBの内容にしたがって，HLレジスタにロード開始アドレス，BCレジスタにデータのバイト数をそれぞれセットしてコールします。このときHLレジスタにセットする値を，こちらで指定するアドレスにしてプログラムを自由にロードすることもできます。LOAD1とLOAD2のルーチンはほとんど同じルーチンですから，使い方に大きな違いはありません。

これらのルーチンを用いて，BASIC CZ 8CB01を3000H以降に，FCBを2040H以降にロードしてみます。リストV-20がそのルーチンです。

ここで大切なことがあります。ロード実行中に，SHIFT＋BREAKやカセットキー，ロ

リストV-20

```
                                ORG     2000H

                        LOAD2   EQU     0044H
                        LOAD1   EQU     0041H
                        CMTCOM  EQU     0DECH
                        PRINT#  EQU     000BH

2000    21 40 20                LD      HL,2040H        ;HL=2040H
2003    01 20 00                LD      BC,0020H        ;BC=0020H
2006    CD 41 00                CALL    LOAD1           ;GOSUB  LOAD1
2009    38 12                   JR      C,ERR?          ;IF Cy=1 GOTO ERR?
200B    21 00 30                LD      HL,3000H        ;HL=3000H
200E    ED 4B 52 20             LD      BC,(2052H)      ;BC=SIZE
2012    CD 44 00                CALL    LOAD2           ;GOSUB  LOAD2
2015    38 06                   JR      C,ERR?          ;IF Cy=1 GOTO ERR?
2017    3E 01           CMTSTP: LD      A,01H           ;Acc=01H
2019    CD EC 0D                CALL    CMTCOM          ;GOSUB  CMTCOM
201C    C9                      RET
                                ;
201D    11 73 14        ERR?:   LD      DE,1473H        ;DE=1473H
2020    CD 0B 00                CALL    PRINT#          ;GOSUB  PRINT#
2023    18 F2                   JR      CMTSTP          ;GOTO CMTSTP
```

モニタ内の文字列

```
:1450=1A FE 61 D8 FE 7B D0 D6 /.0aﾘ0{ﾐﾖ
:1458=20 C9 57 72 69 74 69 6E / ﾉWritin
:1460=67 00 46 6F 75 6E 64 20 /g.Found
:1468=20 00 53 6B 69 70 20 20 / .Skip
:1470=20 00 01 45 52 52 3F 00 / ..ERR?.
:1478=F5 CD 95 10 F1 C3 2D 10 /ﾄﾍ金.ﾄﾃｰ.
```

ードエラーなどで読み込み異常があったとき，キャリーフラグがセットされます。キャリーフラグが立った場合には，ファイルが正しく読み込まれていないことを示しているので，その処理をしなければなりません。

エラーの内容はAccを見て判断できますが，ここでは単にERR? とだけ表示するようにしてみましょう。

エラー処理ルーチンは201DH以降ですが，ここでモニタ内の文字列を利用しています。ダンププコマンドで1458H以降をダンプすると

```
Writing,Found,Skip,ERR?
```

の4つの文字列が見つかります。

モニタはこれら4つのメッセージしか持っていないので，何の異常があってもすべてERR? ですまされてしまいます。

なお，この方法で正しくロードできるのはマシン語ファイルとBASICテキストファイルだけで，ASCIIファイルは正しくロードできません。

ASCIIファイルのデータブロックは，ブロック番号2バイト＋データ256バイトの計258バイトごとに区切られています。そしてテープにFCB，ブロック1，ブロック2……の順に続けて記録されています。

ASCIIファイルのロードには，BCレジスタに0102H(258)をセットし，LOAD2をコールしてください(リストV-21)。ブロック番号はデータの最初にH，Lの順番で書かれています。

ファイルの最後のデータは，ブロック番号がFFFFHになっているので，これを見て最後のデータブロックであることを判断します。

リストV-21

ASCIIファイルのロードルーチン

```
                         ORG     2000H
                 LOAD1   EQU     0041H
                 LOAD2   EQU     0044H
                 CMTCOM  EQU     0DECH

2000  21 40 20           LD      HL,2040H       ;HL=2040H
2003  01 20 00           LD      BC,0020H       ;BC=0020H
2006  CD 41 00           CALL    LOAD1          ;GOSUB  LOAD1
2009  38 1A              JR      C,CMTSTP       ;IF Cy=1 GOTO CMTSTP
200B  21 00 30           LD      HL,3000H       ;HL=3000H
200E  01 02 01           LD      BC,0102H       ;BC=0102H
2011  CD 44 00   LOADLP: CALL    LOAD2          ;GOSUB  LOAD2
2014  38 0F              JR      C,CMTSTP       ;IF Cy=1 GOTO CMTSTP
2016  7E                 LD      A,(HL)         ;Acc=(HL)
2017  FE FF              CP      0FFH           ;IF Acc=0FFH THEN Z=1
2019  20 07              JR      NZ,LDLOP1      ;IF Z<>1 GOTO LDLOP1
201B  23                 INC     HL             ;HL=HL+1
201C  7E                 LD      A,(HL)         ;Acc=(HL)
201D  FE FF              CP      0FFH           ;IF Acc=0FFH THEN Z=1
201F  28 04              JR      Z,CMTSTP       ;IF Z=1 GOTO CMTSTP
2021  2B                 DEC     HL             ;HL=HL-1
2022  09         LDLOP1: ADD     HL,BC          ;HL=HL+BC
2023  18 EC              JR      LOADLP         ;GOTO   LOADLP
2025  3E 01      CMTSTP: LD      A,01H          ;Acc=01H
2027  CD EC 0D           CALL    CMTCOM         ;GOSUB  CMTCOM
202A  C9                 RET
```

ファイルのベリファイをする

ルーチン名　LOAD2？

アドレス　0047H

セーブしたプログラムやデータが，正しくセーブされているかどうかチェックしてみましょう。

BASICのコマンドでは，VERIFYに相当します。このルーチンは，HLレジスタに比較したいデータの先頭アドレスを，BCレジスタにバイト数をセットしてコールします。すると読み込んだデータとメモリに記憶されている値とを比較し，誤りがあった場合や何かの異常があった場合は，キャリーフラグがセットされエラーの種類がAccにセットされます。

リストV-22

```
                              ORG     2000H

                    LOAD1     EQU     0041H
                    LOAD2?    EQU     0047H
                    PRINT#    EQU     000BH
                    CMTCOM    EQU     0DECH

2000   21 40 20               LD      HL,2040H          ;HL=2040H
2003   01 20 00               LD      BC,0020H          ;BC=0020H
2006   CD 41 00               CALL    LOAD1             ;GOSUB  LOAD1
2009   38 0C                  JR      C,CHECK           ;IF Cy=1 GOTO CHECK
200B   21 00 30               LD      HL,3000H          ;HL=3000H
200E   ED 4B 52 20            LD      BC,(2052H)        ;BC=SIZE
2012   CD 47 00               CALL    LOAD2?            ;GOSUB  LOAD2?
2015   30 22                  JR      NC,CMTSTP         ;IF Cy=0 GOTO CMTSTP
2017   FE 01        CHECK:    CP      01H               ;IF Acc=01H THEN Z=1
2019   11 60 20               LD      DE,MSSG1          ;DE=MSSG1
201C   28 18                  JR      Z,PRTMSG          ;IF Z=1 GOTO PRTMSG
201E   FE 02                  CP      02H               ;IF Acc=02H THEN Z=1
2020   21 66 20               LD      DE,MSSG2          ;DE=MSSG2
2023   28 11                  JR      Z,PRTMSG          ;IF Z=1 GOTO PRTMSG
2025   FE 03                  CP      03H               ;IF Acc=03H THEN Z=1
2027   11 76 20               LD      DE,MSSG3          ;DE=MSSG3
202A   28 0A                  JR      Z,PRTMSG          ;IF Z=1 GOTO PRTMSG
202C   FE 04                  CP      04H               ;IF Acc=04H THEN Z=1
202E   11 82 20               LD      DE,MSSG4          ;DE=MSSG4
2031   28 03                  JR      Z,PRTMSG          ;IF Z=1 GOTO PRTMSG
2033   11 90 20               LD      DE,MSSG5          ;DE=MSSG5
2036   CD 0B 00     PRTMSG:   CALL    PRINT#            ;GOSUB  PRINT#
2039   3E 01        CMTSTP:   LD      A,01H             ;Acc=01H
203B   CD EC 0D               CALL    CMTCOM            ;GOSUB  CMTCOM
203E   C9                     RET
                              ;
                              ORG     2060H
2060   42 72 65 61  MSSG1:    DB      "Break",00H
2064   6B 00
2066   43 68 65 63  MSSG2:    DB      "Check Sum Error",00H
206A   6B 20 53 75
206E   6D 20 45 72
2072   72 6F 72 00
2076   4F 75 74 20  MSSG3:    DB      "Out of Tape",00H
207A   6F 66 20 54
207E   61 70 65 00
2082   57 72 69 74  MSSG4:    DB      "Write Protect",00H
2086   65 20 50 72
208A   6F 74 65 63
208E   74 00
2090   54 61 70 65  MSSG5:    DB      "Tape End",00H
2094   20 45 6E 64
2098   00
```

リストV-23

```
:2000=21 40 20 01 20 00 CD 41 /!@ . .^A
:2008=00 38 0C 21 00 30 ED 4B /.8.!.0┘K
:2010=52 20 CD 47 00 30 22 FE /R ^G.0
:2018=01 11 60 20 28 18 FE 02 /..` (.o.
:2020=11 66 20 28 11 FE 03 11 /.f (.o..
:2028=76 20 28 0A FE 04 11 82 /v (.o..Σ
:2030=20 28 03 11 90 20 CD 0B / (..● ^.
:2038=00 3E 01 CD EC 0D C9 00 /.>.^ﾞ.ﾉ.
:2040=00 00 00 00 00 00 00 00 /........
:2048=00 00 00 00 00 00 00 00 /........
:2050=00 00 00 00 00 00 00 00 /........      ベリファイルーチン
:2058=00 00 00 00 00 00 00 00 /........      (リストV-22)
:2060=42 72 65 61 6B 00 43 68 /Break.Ch
:2068=65 63 6B 20 53 75 6D 20 /eck Sum
:2070=45 72 72 6F 72 00 4F 75 /Error.Ou
:2078=74 20 6F 66 20 54 61 70 /t of Tap
:2080=65 00 57 72 69 74 65 20 /e.Write
:2088=50 72 6F 74 65 63 74 00 /Protect.
:2090=54 61 70 65 20 45 6E 64 /Tape End
:2098=00 00 00 00 00 00 00 00 /........
:20A0=21 40 20 01 20 00 CD 41 /!@ . .^A
:20A8=00 38 0C 21 00 30 ED 4B /.8.!.0┘K
:20B0=52 20 CD 44 00 30 22 FE /R ^D.0
:20B8=01 11 60 20 28 18 FE 02 /..` (.o.
:20C0=11 66 20 28 11 FE 03 11 /.f (.o..      ロードルーチン
:20C8=76 20 28 0A FE 04 11 82 /v (.o..Σ
:20D0=20 28 03 11 90 20 CD 0B / (..● ^.
:20D8=00 3E 01 CD EC 0D C9 00 /.>.^ﾞ.ﾉ.
:20E0=21 40 20 01 20 00 CD 3B /!@ . .^;
:20E8=00 38 0C 21 00 30 ED 4B /.8.!.0┘K
:20F0=52 20 CD 3E 00 30 22 FE /R ^>.0
:20F8=01 11 60 20 28 18 FE 02 /..` (.o.      セーブルーチン
:2100=11 66 20 28 11 FE 03 11 /.f (.o..
:2108=76 20 28 0A FE 04 11 82 /v (.o..Σ
:2110=20 28 03 11 90 20 CD 0B / (..● ^.
:2118=00 3E 01 CD EC 0D C9 00 /.>.^ﾞ.ﾉ.
```

使い方はLOAD2とよく似ています。リスト20でCALL LOAD2としているところをCALL LOAD2？　とすると，ベリファイルーチンになります。しかし，ここではさらに，Accの内容を見て，何のエラーがあったのかを判断してみましょう(リストV-22)。

リストV-23は，リストV-22のベリファイルーチン(2000H～2098H)のダンプリストですが，20A0H～20DEHにはロードルーチン，20E0H～211EHにはあとで説明するセーブルーチンとなっています。

これらの3つのルーチンは，ベリファイルーチンのCALL LOAD1とCALL LOAD2?に対応するところを図V-8に示すように変更する以外はまったく同じです。

図V-8　3つのルーチンの変更個所

ベリファイルーチン	ロードルーチン	セーブルーチン
CALL　LOAD 1	CALL　LOAD 1	CALL　SAVE 1
CALL　LOAD 2 ?	CALL　LOAD 2	CALL　SAVE 2

リストV-24

書き換え前

```
:3F40=00 00 06 46 49 4C 45 53 /...FILES
:3F48=0D 00 00 00 00 00 00 00 /........
:3F50=00 00 07 3F 54 49 4D 45 /...?TIME
:3F58=24 0D 00 00 00 00 00 00 /$.......
:3F60=00 00 03 4B 45 59 00 00 /...KEY..
:3F68=00 00 00 00 00 00 00 00 /........
:3F70=00 00 06 4C 49 53 54 1A /...LIST.
:3F78=0D 00 00 00 00 00 00 00 /........
:3F80=00 00 06 52 55 4E 20 20 /...RUN
:3F88=0D 00 00 00 00 00 00 00 /........
:3F90=00 00 06 4C 4F 41 44 20 /...LOAD
:3F98=0D 00 00 00 00 00 00 00 /........
:3FA0=00 00 06 57 49 44 54 48 /...WIDTH
:3FA8=20 00 00 00 00 00 00 00 / .......
:3FB0=00 00 05 43 48 52 24 28 /...CHR$(
:3FB8=00 00 00 00 00 00 00 00 /........
:3FC0=00 00 06 50 41 4C 45 54 /...PALET
:3FC8=20 00 00 00 00 00 00 00 / .......
:3FD0=00 00 05 43 4F 4E 54 0D /...CONT.
:3FD8=00 00 00 00 00 00 00 00 /.......
```

書き換え後

```
:3F40=00 00 04 52 55 4E 0D 00 /...RUN..
:3F48=00 00 00 00 00 00 00 00 /........
:3F50=00 00 04 4C 49 53 54 00 /...LIST.
:3F58=00 0D 00 00 00 00 00 00 /........
:3F60=00 00 04 41 55 54 4F 00 /...AUTO.
:3F68=00 00 00 00 00 00 00 00 /........
:3F70=00 00 05 52 45 4E 55 4D /...RENUM
:3F78=00 00 00 00 00 00 00 00 /........
:3F80=00 00 05 43 4F 4C 4F 52 /...COLOR
:3F88=00 00 00 00 00 00 00 00 /.......
:3F90=00 00 05 43 48 52 24 28 /...CHR$(
:3F98=00 00 00 00 00 00 00 00 /........
:3FA0=00 00 08 44 45 46 20 4B /...DEF K
:3FA8=45 59 28 00 00 00 00 00 /EY(.....
:3FB0=00 00 04 43 4F 4E 54 00 /...CONT.
:3FB8=00 00 00 00 00 00 00 00 /........
:3FC0=00 00 04 53 41 56 45 00 /...SAVE.
:3FC8=00 00 00 00 00 00 00 00 /.......
:3FD0=00 00 04 4C 4F 41 44 00 /...LOAD.
:3FD8=00 00 00 00 00 00 00 00 /........
```

リストV-23をモニタから打ち込んでください。この3つのルーチンを使ってCZ-8CB01のバックアップをとりベリファイをしてみましょう。

CZ-8CB01のテープをセットし，20A0Hから実行するプログラムの本体は3000H以降にロードされます。

新しいテープをセットして20E0Hを実行すると，CZ-8CB01がセーブされますが，セーブするのは少し待ってください。セーブする前にCZ-8CB01に少し手を加えてみます。

例として，ファンクションキーの内容を書き換え，PRINT文の中にカーソルコントロールを取り入れることのできるBASICにしてみましょう(リストV-24)。

IOCSの0F42H～0FE1H(160バイト)は，ファンクションキーの定義状態を示すファン

クションキーバッファになっています。この内容を書き換えれば，ファンクションキーに自由な値を定義できるのですが，ファンクションキーバッファの構造は，ひとつのキーに16バイトが当てられており1バイト目は定義文字数で残り15バイトは定義文字が入ります。F1キーは0F42Hから F2 は0F52Hから始まります。F1を見てみると，0F42Hの最初の05が定義文字数ですから，0DまでがF1のファンクションキーの内容であることがわかります(ここで0Dは，CR⏎のコードです)。

実際にCZ-8CB01を書き換えてみましょう。注意することは，CZ-8CB01の本体は0000Hからロードされるのですが，ここでは3000H以降に置かれているので，0F42H～0FE1Hまでに相当するアドレスは，3F42H～3FE1Hになっているということです。

リストV-24の例を参考に，自分の好きなようにファンクションキーを定義してみてください。また01A2H(31A2H)をB7H(OR A)に書き換えると ESC ＋コントロールコードの入力で，カーソルコントロールなどのコントロールコードが出力できます。これをPRINTの中に入れて，カーソル制御の表示もできるようになります(図V-9)。

これらの改訂を加えたCZ-8CB01を，

```
*G20E0 ⏎
```

としてセーブしてください。テープを巻き戻して，

```
*G2000 ⏎
```

とするとベリファイが始まり，異常があればそのメッセージが出力されます。途中でカセットキーを押すなどしていろいろ試してみてください。

ファイルコントロールブロックのセーブ

ルーチン名　SAVE1

アドレス　　003BH

FCBの内容を自由に書き換えられるようになると，こんどは次のようなことができます。

① プログラムを，実行させるアドレスとまったく別の場所に作り，テープにセーブして IPL から実行させる。

② BASIC インタプリタなどのようなマシン語プログラムを改訂したり，使えそうなルーチンを分離したりジョイントする。

③ 普通にはありえないファイルモードを作り，BASIC, IPL からロードできないファイルにして内容を秘密化する。

①および②で注意することは，FCB に書かれているプログラムのサイズと，データブロックの長さを合わせておかないと，IPL や BASIC からロードしたときチェックサムエラーが出たり，正しくロードできなかったりします。SAVE1 をコールするときに BC レジスタにセットする値は，FCB の 19, 20 バイト目に書かれているプログラムのサイズと同じ値をセットしてください。

BASIC CZ-8CB01 の 0000H～149FH まではIOCS とモニタになっています。この部分は役に立つルーチンをたくさん含んでいます。また，BASIC から分離して使用できますので，ここではそれを分離してみます。

Lコマンドを用いて CZ-8CB01 を 3000H からロードします。

リストV-25

```
:3118=CD 26 0B ED 5E 21 46 03 /ヘ&.┘^!F.
:3120=22 52 00 CD 2D 01 CD 3C /'RUN-0!<
:3128=01 FB C3 A0 14 21 52 00 /.ｼﾃﾂ.!R.
   A0 14 → E2 0F


:4040=D5 D9 C9 44 8B 11 4D 1D /ユルノD◣.M.

:4048=12 50 61 10 47 86 11 46 /.Pa.G♦.F

:4050=53 12 52 F0 10 53 6A 10 /S.R°.Sj.

:4058=4C 9A 10 56 E1 10 54 BE /Lα.V| .Tセ

:4060=12 3A 72 14 EE 01 32 72 /.:r.└.2r

:4068=14 C9 CD A6 12 D8 ED 43 /.ノヘヲ.リ┘C
 F0 10→FA 00


:2020=01 58 2D 31 20 4D 6F 6E /.X-1 Mon
:2028=69 74 6F 72 20 20 20 20 /itor
:2030=20 20 A0 14 00 00 00 00 /   .....
:2038=00 00 00 00 00 00 00 00 /........
```

リスト V-25 に従って変更してください。変更したところの意味は，次のとおりです。

312BH，312CH（実アドレス012BH，012CH）

BASIC のホットスタートのアドレスです。モニタのコールドスタートに変更(0FE2H)。

4053H，4054H（実アドレス1053H，1054H）

モニタのＲコマンドのジャンプアドレスです。システムイニシャライズに変更(00FAH)。

2020Hから32バイト

FCB です。CZ-8CB01 で使われている形式に従って作っていきます。

2032Hから２バイト

データのバイト数です。モニタと IOCS を合わせると 14A0H バイトになります。

2034Hから２バイト

ロード開始アドレスです。ここを 0000H とすると，IPL からロードするときには 0000H からロードされます。

2036Hから２バイト

オートスタートアドレスです。IPL からロードした場合，このアドレスから実行されます。ここでは 0000H にセットします。

この FCB をセーブするには BC レジスタに 0020H，HL レジスタに 2020H をセットして SAVE1 をコールすると FCB のセーブが始まります。

新しいテープをセットし，リスト V-26 を実行すると，まず FCB がセーブされ，続いて 3000H～449FH までのデータブロックがセーブされます。これを IPL からロードすると，FCB の内容に従って 0000H からロードされます。

このモニタは，IOCS を含む立派なカセット・オペレーティング・システムとしての実力を持っているのでいろいろ応用ができます。今まで紹介してきたルーチンは，すべてこのモニタから実行できるので試してみてください。

リスト V-26

```
                                  ORG     2000H

                         SAVE1    EQU     003BH
                         SAVE2    EQU     003EH
                         CMTCOM   EQU     0DECH

2000    21 20 20                  LD      HL,2020H
2003    01 20 00                  LD      BC,0020H
2006    CD 3B 00                  CALL    SAVE1
2009    21 00 30                  LD      HL,3000H
200C    ED 4B 32 20               LD      BC,(2032H)
2010    CD 3E 00                  CALL    SAVE2
2013    3E 01                     LD      A,01H
2015    CD EC 0D                  CALL    CMTCOM
2018    C9                        RET
```

データブロックをセーブする

ルーチン名　SAVE2

アドレス　　003EH

リスト V-26 にすでに使われましたが，SAVE1 と使い方は同じで，こちらはデータブロックをセーブするルーチンです。HL レジスタにセーブしたいデータの先頭アドレス，BC レジスタにバイト数をセットしてコールします。

実際にデータがロードされる場合，FCB の内容によってロードされるアドレスが決まるので 3000H からセーブしたプログラムを 0000H からロードさせて実行することもできます。また FCB をつけずにデータブロックだけセーブすることもできます。

とくに後者は，BASIC では ASCII セーブや DEVO$ “CAS:” で使われています。このようなセーブの仕方をブロッキングセーブといい，1 レコード(256 バイト)単位で行われます。カセットの場合は，これにレコード番号 2 バイトを加えた 258 バイト単位でセーブが行われます。

このルーチンの使い方の例として，リスト V-21 でロードした ASCII ファイルのセーブルーチンを作ってみます。リスト V-27 がそれです。

リスト V-21 と重ならないよう 2060H から作ります。実行は 2060H からです。このルーチンは 2066H で FCB をセーブしたあと，2071H から 1 データブロックをセーブしています。

レコード番号は 258 バイトごとに現れるので，これが FFFFH になるまでデータブロックのセーブを繰り返します。

SAVE2 ルーチンの実行中に何かのエラーがあった場合 LOAD1，LOAD2，LOAD2?，SAVE1 と同様，Acc にエラー内容がセットされ，キャリーフラグがセットされます。リスト V-27 ではエラーの種類までは表示されませんので，工夫してより完全なルーチンに仕上げてみてください。

リスト V-27

```
                            ORG     2060H

                   SAVE1    EQU     003BH
                   SAVE2    EQU     003EH
                   CMTCOM   EQU     0DECH

2060    21 40 20            LD      HL,2040H        ;HL=2040H
2063    01 20 00            LD      BC,0020H        ;BC=0020H
2066    CD 3B 00            CALL    SAVE1           ;GOSUB  SAVE1
2069    38 1A               JR      C,CMTSTP        ;IF Cy=1 GOTO CMTSTP
206B    21 00 30            LD      HL,3000H        ;HL=3000H
206E    01 02 01            LD      BC,0102H        ;BC=0102H
2071    CD 3E 00   ASCLPO:  CALL    SAVE2           ;GOSUB  SAVE2
2074    38 0F               JR      C,CMTSTP        ;IF Cy=1 GOTO CMTSTP
2076    7E                  LD      A,(HL)          ;A=(HL)
2077    FE FF               CP      0FFH            ;IF A=FFH THEN Z=1
2079    20 07               JR      NZ,ASVLOP       :IF Z=0 GOTO ASVLOP
207B    23                  INC     HL              ;HL=HL+1
207C    7E                  LD      A,(HL)          ;A=(HL)
207D    FE FF               CP      0FFH            ;IF A=FFH THEN Z=1
207F    28 04               JR      Z,CMTSTP        ;IF Z=1 GOTO CMTSTP
2081    2B                  DEC     HL              ;HL=HL-1
2082    09         ASVLOP:  ADD     HL,BC           ;HL=HL+BC
2083    18 EC               JR      ASCLPO
2085    3E 01      CMTSTP:  LD      A,01H           ;A=01H
2087    CD EC 0D            CALL    CMTCOM          ;GOSUB  CMTCOM
208A    C9                  RET
```

IOCSワークエリアの働きと操作法

主要なサブルーチンの利用方法をみてきましたが，IOCS内ルーチンを利用する場合は，ワークエリアの働きが重要になります。

ワークエリアとは，システムの動作に必要な各種パラメータを設定あるいは格納するメモリアドレスです。システムがこれらの値を参照して動作し，それに応じてワークエリアの値を更新します。

ここではIOCS内のワークエリアのそれぞれの働きと操作方法をアドレス順にみていきます。

LINELIM (0006H)

このアドレスの値によってINPUTF/LININP/BINPUTで使われるキー入力バッファの大きさが制限されます。この値を大きくとると，キー入力バッファとして確保しなければならないエリアが大きくなります。

BASICではFFHにセットされているので255文字までは1行として読み込むことができます。モニタでは50H(80バイト)にセットされています。したがってキーバッファに読み込まれる文字数は80文字までで，それ以上の文字は無視されます。

モニタのキー入力バッファの先頭はFF00Hなので，あまり大きな値に書き換えると，FFFFHから積まれているスタックエリアを壊してしまいます。プログラムの中でLINELIMを操作した場合，モニタに帰る直前に50Hにセットしてください。

WIDTH 0 (0007H)

現在のスクリーンモード(WIDTH40/WIDTH80)を示しており，28H(10進数=40)から50H(10進数=80)の値が書かれています。このワークエリアは読むだけに使い，書き換えてはいけません。

CURX (000EH)

CURY (000FH)

CURXとCURYは，それぞれ現在あるカーソル位置のX座標とY座標を示しています。書き換えると，カーソルの位置が変更できます。

CURXとCURYに値をセットし，ADRCAL(054AH)をコールすると，カーソル位置のテキストV-RAMのアドレスがHLレジスタにセットされて帰ってきます。

リストV-28を実行すると画面の中央に『A』が表示されます。モニタからメモリセットコマンドでA001Hアドレスを書き終えると，カーソルの位置が変化するのがわかります。ただしX座標は⏎するごとに左端に移ることを，リストV-28のルーチンを実行して試し

てください。

```
リストV-28
                                    ORG     0A000H

                            ACCDIS  EQU     0013H

A000    3E 14                       LD      A,14H
A002    32 0E 00                    LD      (000EH),A
A005    3E 41                       LD      A,41H
A007    CD 13 00                    CALL    ACCDIS
A00A    C9                          RET
```

CURYST(0016H)

CURYED(0017H)

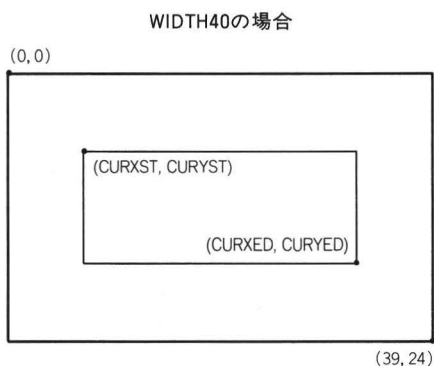
CURXST(001EH)

CURXED(001FH)

これらのワークエリアは，BASICのCONSOLEに相当し，書き換えることによって画面のテキストのウィンドウを制限することができます。

これらのワークエリアの関係は，図V-10のようになります。

このエリアはWIDTH48(004DH)をコールすることにより最大値がセットされます。

図V-10 CURXST, CURYST, CURXED, CURYEDの画面関係

COLORF(0026H)

キャラクタのアトリビュートにセットする値を示しています。アトリビュートの各ビットの意味は，次のようになっています。

bit	0	青	COLOR	色の指定
bit	1	赤		
bit	2	緑		
bit	3	CREV		反転文字
bit	4	CFLASH		点滅文字

bit　5　CGEN　　　CG/PCG
bit　6 ⎫
　　　 ⎬ CSIZE　　キャラクタサイズ ⎧ 縦2倍
bit　7 ⎭　　　　　　　　　　　　 ⎩ 横2倍

このワークエリアは，通常07H(白)にセットされています。書き換えるとそれ以降に実行されるAccDISやPRINT#などで表示されるキャラクタのアトリビュートに，すべてこの値が書き込まれます。

メモリセットコマンドで，いろいろに書き換えて試してみてください。

KEYDAT(002EH〜002FH)

このワークエリアには，現在押されているキーの情報が書かれています。

キーデータは2バイトで構成されますが，002EHにASCIIコード，002FHにファンクションキーデータが入っています。したがってリアルタイムキースキャンをしたいときにはこのワークエリアを調べます。

何もキーが押されていないと002EHにNULコード，002FHにFFHがセットされます。

BRKBUF(0036H)

このワークエリアは割り込みによるキー入力で，[SHIFT]＋[BREAK]および[BREAK]が押された場合，03Hまたは13Hが書き込まれます。

このワークエリアはBREAK処理が行われると，すぐにクリアされるので通常は00Hが入っています。

KEYFLG(0037H)

割り込みによるキー入力処理は，TVキー，ボリュームキーなどが押された場合や，キーが離された場合も割り込みがかかります。無効なキーが割り込んだ場合には，このワークエリアが00Hになります。このエリアは読むだけで書き込みはできません。

INIARR(00E9/00EAH)

WIDTH40のとき，現在表示している画面のオフセット値が入ります。SCREEN0のV-RAMの開始アドレスは3000H，SCREEN1のV-RAMの開始アドレスは，0400H番地あとの3400Hから始まります。この0400Hの差がオフセット値です。

00E9/00EAHには，このオフセット値がH,Lの順で書かれています。このワークエリアは書き換えてはいけません。

INIADW(00EB/00ECH)

これから書き込もうとしている画面のオフセット値が入ります。オフセット値は00EB/00ECHにL,Hの順に書き込まれています。

このワークエリアも書き換えてはいけません。

INBUF(0EA8H～0EE7H)

POINT 1(0EA6H)

POINT 2(0EA7H)

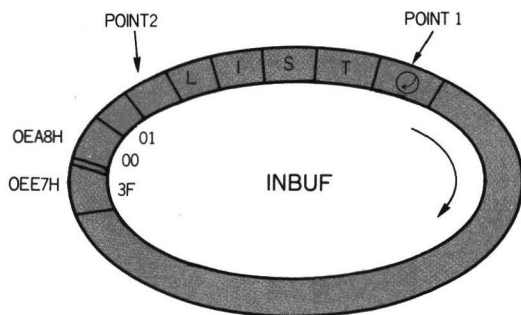
Hu-BASICの特徴である先行入力バッファのワークエリアです。このバッファはリングバッファになっていて，64バイトまでの文字を蓄えることができます。

先行入力バッファというのは，CPUがほかの処理で忙しいとき，割り込みによるキー入力処理で入力されたキーデータを一時このバッファにためておき，キーデータが必要になってから読み出します。

IOCS内にあるもので，80C49内にあるキーバッファとは別のものです。

POINT1とPOINT2は，それぞれこのバッファの書き込みポインタと，読み出しポインタを示しています。割り込みキー入力があると次々にこのバッファに書き込まれ，POINT1が進んでいきます。

このバッファはリング状に接続されているので，0EE7Hの次は0EA8Hに書き込まれます。POINT1がPOINT2の直前にくると，このバッファはいっぱいになり，それ以上の入力は受けつけません(図V-11)。

図V-11　キー入力リングバッファの構造図

INPUTF/BINPUT/INKEY$などのキー入力ルーチンをコールすると，このバッファからデータをとり込みPOINT2を進めていきます。POINT1とPOINT2の値が同じになるとバッファは空になります。

Hu-BASICのKEY0文では，このバッファにデータをセットすることで先行入力を設定しています。

例としてキーバッファにRUNをセットし，ロード終了後オートランするBASICを作成してみましょう。再びリストV-23を用います。

CZ-8CB01をセットし，20A0Hから実行してロードし，次に先行入力を設定します。メモリセットコマンドで次のように設定してください。

```
*M3EA6 ⏎
:3EA6=0B ⏎
:3EA7=00 ⏎
:3EA8=00 ;A ;P ;S ;S ;1 ;: ;R ;U ;N 0D ⏎
```

新しいテープを用意し，20E0H から実行して CZ-8CB01 をセーブしてください。そのあとに BASIC のプログラムをセーブしておくと，CZ-8CB01 のロード終了後自動的にロードが始まり RUN します。

TABBUF (0EF2H～0F41H)

水平タブの位置を設定するバッファで，80 バイトの大きさを持っています。

タブを設定する場合には，対応するバッファに 00H 以外の値を書き込みます。

FUNBUF (0F42H～0FE1H)

このワークエリアには，ファンクションキーの定義内容が書かれています。ファンクションキーのバッファは，16 バイトずつ 10 個分のエリアが確保されています。16 バイトのうち先頭の 1 バイトが，定義されている文字のバイト数で残り 15 バイトに定義文字が入ります。

F1 のファンクションキーバッファを例にとれば，図V-12 のような構造になっています。

図V-12　F1 ファンクションキーバッファデータ構造

0F42H ～ 0F51H

05	A	U	T	O	0D	00	00	00	00	00	00	00	00	00	00

Hu-BASIC では，ファンクションキーの初期値は図V-13 のとおりです。

図V-13　ファンクションキーの初期値

ファンクションキーナンバー	バイト数	定義内容
1	5	"AUTO" + CHR$(13)
2	7	"?TIME$" + CHR$(13)
3	3	"KEY"
4	6	"LIST" + CHR$(26,13)
5	6	"RUN␣␣" + CHR$(13)
6	6	"LOAD␣" + CHR$(13)
7	6	"WIDTH␣"
8	5	"CHR$("
9	6	"PALET␣"
10	5	"CONT" + CHR$(13)

このバッファの内容を書き換えると，ファンクションキーの定義が変わります。

IOCSルーチン表(X1/C/D/F)

アドレス	ルーチン名	レジスタ	説明
0000H 0066H	START NMI	破壊 ——— 入力 ——— 出力 ———	システムイニシャライズを行うコールドスタート
0003H	INPUTF	破壊 AF 入力 DE:データ先頭アドレス 出力 DE:入力データ先頭アドレス,異常のときCy=1,Acc:異常コード	1行分のスクリーンエディットの実行 DEレジスタで指定されたアドレスに1行分のデータを取り込む
000BH	PRINT#	破壊 AF 入力 DE:文字列の先頭アドレス 出力 ———	DEで示すアドレスから始まる文字列(終わり00H)を画面に出力
0013H	Acc PRT	破壊 ——— 入力 Acc:出力するASCIIコード 出力 ———	Accで示すASCIIコードの文字を画面に1文字出力
001BH	INKEY$	破壊 AF 入力 Acc:INKEY$のモード 出力 Acc:キーボードから入力した文字のASCIIコード	キー入力モード:①FFH…INKEY$(キーバッファ有効), ②01H…INKEY$(1), ③02H…INKEY$(2), ④00H…INKEY$(0)(リアルタイム)
0023H	TAK49S	破壊 AF, BC, DE 入力 Acc:80C49に送るコマンド, DE:データバッファ先頭アドレス 出力 Acc:(0:OUT/1:IN), C:送受したデータのバイト数	サブCPU80C49との通信, Accにセットされたコマンドにしたがい後続データの送受を自動的に行う
002BH	CGSET	破壊 AF, E, HL 入力 D:CHRコード(00~FFH), E:セットI/Oアドレス上位(15~17H), HL:セットするデータの先頭アドレス 出力 HL:INの値より+8される	PCGのセット
0033H	CGREAD	破壊 AF, E, HL 入力 D:CHRコード(00~FFH), E:リードI/Oアドレス上位(14~17H), HL:データバッファの先頭アドレス 出力 HL:INの値より+8される	PCG, CGROMのドットパターンのメモリ上への読み出し E:14H…ROMCG, 15H…RAMCG BLUE, 16H…RAMCG RED, 17H…RAMCG GREEN
003BH	SAVE1	破壊 AF 入力 HL:FCBの先頭アドレス, BC:FCBの長さ(20H) 出力 Acc:異常コード(00~05H), Cy1:異常	ファイルコントロールブロックのカセットテープへの記録 異常コードAcc(注1参照)
003EH	SAVE2	破壊 AF 入力 HL:セーブデータの先頭アドレス, BC:セーブデータのバイト数 出力 Acc:異常コード(00~05H), Cy1:異常	データブロックのカセットテープへの記録 異常コードAcc(注1参照)
0041H	LOAD1	破壊 AF 入力 HL:FCBロード先頭アドレス, BC:FCBの長さ 出力 Acc:異常コード, Cy1:異常	HLの示すアドレスへのファイルコントロールブロック, BCの長さのロード 異常コードAcc(注1参照)
0044H	LOAD2	破壊 AF 入力 HL:データロード先頭アドレス, BC:データの長さ 出力 Acc:異常コード, Cy1:異常	HLの示すアドレスへのデータブロック, BCの長さのロード 異常コードAcc(注1参照)
0047H	LOAD2 ?	破壊 AF 入力 HL:チェック先頭アドレス, BC:比較バイト数 出力 Acc:異常コード, Cy1:異常	HLの示すアドレスからBCバイトのベリファイ 異常コードAcc(注1参照)
004AH	BRKCHK	破壊 AF 入力 ——— 出力 Z=1:BREAK	ルーチン実行中にブレークキーが押されたかどうかのチェック
004DH	WIDTH48	破壊 AF, BC, DE, HL 入力 Acc:横の文字数をセット(40/80) 出力 ———	CRTCにAcc≦40(29H)なら40文字モード, Acc>40(29H)なら80文字モードの指定をし, ワークWIDTH0(0007H)に40か80をセット(このときテキスト, グラフィックのエリア, コンソール解除をする。ワークSCRMDD(0A8BH)が2ならグラフィックはクリアしない)。
012DH	KVECIN	破壊 HL, AF, I 入力 ——— 出力 I:キー割り込みベクトル上位	ワークエリアTNITTAB0052, 0053Hにキー入力割り込み処理アドレスをセットしてCALLするとIレジスタにベクトル(アドレス)上位を設定, 下位を80C49に伝える
0133H	KVEC 00	破壊 AF 入力 L:キー割り込みベクトル下位 出力 ———	80C49にキー入力割り込みベクトルの下位を伝達 Lレジスタに00Hを入れるとキー割り込み解除
0346H	INTKEY	破壊 ——— 入力 ——— 出力 ———	割り込みキー入力の実行, キーデータはワーク"KEYDAT"(002E/002FH)にセット 002EH:ASCIIコード, 002FH:ファンクションデータ
04A7H	CR1	破壊 AF 入力 ——— 出力 ———	改行

アドレス	ルーチン名	レ ジ ス タ	説 明
04A3H	CR2	破壊 AF 入力 ——— 出力 ———	現在のカーソル位置が行の先頭でないなら改行
04ABH	TABPRT	破壊 AF 入力 ——— 出力 ———	X座標位置が10の倍数になるまでのスペースの出力
04BAH	SPPRT	破壊 AF 入力 ——— 出力 ———	スペース1個出力
04C8H	AccDIS	破壊 ——— 入力 Acc:出力する文字のASCIIコード 出力 ———	00〜1FHのコントロールコードもキャラクタとして表示，それ以外はAcc PRTと同じ
054AH	ADRCAL	破壊 AF, BC, HL 入力 ——— 出力 カーソル位置のテキストV-RAMアドレス	現在のカーソル位置に対応するテキストV-RAMのアドレスをHLにセット アトリビュートV-RAMアドレスはHL－1000Hで求める
054DH	ADRCA2	破壊 AF, BC, HL 入力 H:テキスト座標Yの値，L:テキスト座標Xの値 出力 HL:テキストV-RAMのアドレス	H,Lにセットされた画面のX,Y座標に対応するテキスト V-RAMのアドレスをHLにセット，アトリビュートV-RAMアドレスはHL－1000Hで求める
0577H	CTRLJB	破壊 AF, BC, DE, HL 入力 Acc:コントロールコード(00〜1FH) 出力 ———	Accで指定されたコントロールコードの実行
07F7H	BEEP	破壊 AF, BC, HL, D 入力 ——— 出力 ———	短い音の出力，BASICのBEEPと同じ
098CH	WIDTH80	破壊 AF, BC, DE, HL 入力 ——— 出力 ———	WIDTH80の実行，CRTCに80文字のモードを指定 BASIC WIDTH80
0998H	WIDTH40	破壊 AF, BC, DE, HL 入力 ——— 出力 ———	WIDTH40の実行，CRTCに40文字のモードを指定 BASIC WIDTH40
09C0H	SCRNOT	破壊 AF 入力 Acc:出力ページ(00/01H) 出力 ———	Accの示すページを出力ページに設定
09F5H	SCRNIN	破壊 AF 入力 Acc:書き込みページ(0か1) 出力 ———	Accの示すページを書き込みページに設定
0A3FH	CTRLD ?	破壊 AF 入力 ——— 出力 ———	CONSOLEを解除し，入出力ページを両方とも0ページに設定
0A5AH	STPRIO	破壊 AE, BC, D, HL 入力 ——— 出力 ———	プライオリティフラグ(00F6〜00F9H)の値にしたがってプライオリティをセット
0A6BH	STCLST	破壊 AE, BC, D, HL 入力 ——— 出力 ———	テキスト画面のクリア
0A8AH	STCLSG	破壊 AF, BC 入力 ——— 出力 ———	グラフィックのオールクリア(ただしワークSCRMOD・0A8BHの値が02Hなら実行しない)
0B49H	RECV49	破壊 AF 入力 ——— 出力 Acc:サブCPU80C49から受けとったデータ	サブCPUからデータが送られるまで待ち，1バイト受信
0B54H	TRANS49	破壊 AF 入力 Acc:80C49に送るデータ 出力 ———	サブCPU80C49がデータを受け取れる状態になるまで待ち，1バイト送信
0B61H	OBFCK	破壊 AF, BC 入力 ——— 出力 ———	サブCPU80C49のOBFをチェックし，80C49がデータを受けとれる状態になるまでウェイト
0B6BH	IBFCK	破壊 AF, BC 入力 ——— 出力 ———	サブCPU80C49のIBFをチェックし，80C49からデータが送られてくるまでウェイト

アドレス	ルーチン名	レ ジ ス タ	説 明
0C8AH	WBYTE	破壊 AF 入力 Acc:テープに書く1バイトのデータ 出力 ———	Accの値のカセットへの書き込み
0CAAH	RBYTE	破壊 AF 入力 ——— 出力 Acc:テープから読んだ1バイトデータ	カセットからAccへの1バイト読み込み
0CD6H	CLRSUM	破壊 ——— 入力 ——— 出力 ———	チェックサムバッファ(0EA1H)のクリア
0CDFH	GAP	破壊 AF, DE 入力 ——— 出力 ———	カセットへのリーダ音の書き込み
0D20H	TMARK	破壊 AF, DE 入力 ——— 出力 ———	カセットのリーダ音待ち
0D8AH	SHORT	破壊 ——— 入力 ——— 出力 ———	0のカセットへの1ビット書き込み
0DA4H	LONG	破壊 ——— 入力 ——— 出力 ———	1のカセットへの1ビット書き込み
0DC7H	MOTOR	破壊 Acc 入力 D:00/04/08/0CH…READ, その他…WRITE 出力 Acc:OUT OF TAPE…03H, WRITE PROTECT…04H, Cy=1:異常	カセットの読み書き開始(Dの値が04Hの倍数はREAD, それ以外WRITE)
0DECH	CMTCOM	破壊 AF 入力 Acc:カセットコマンド 出力 ———	Accに示すカセットのコマンドの実行, セット値はBASICのCMTコマンドに同じ(WRITEは0AH)Acc:00…EJECT, 01…STOP, 02…PLAY, 03…FF, 04…REW, 05…APSS+1, 06…APSS−1, 0A…WRITE
0DF6H	CMTSNS	破壊 AF 入力 ——— 出力 Acc:カセットの状態	カセットの状態をAccにセット Acc:bit0…0ならTAPE END, bit2…0ならOUT OF TAPE bit2…0ならWRITE PROTECT
0DFEH	COMOUT	破壊 AF 入力 Acc:80C49に送るコマンド 出力 ———	サブCPU80C49へのAccのコマンド送信(コマンドについては80C49コマンド表参照), TRANS49またはRECV49をコールする前に実行のこと
0E77H	PSGSET	破壊 AF, BC, D, HL 入力 HL:PSGセットデータバッファの先頭アドレス 出力 HL:+8される	HLで始まる8バイトのデータをPSGにセット (レジスタアドレス:00~0FH) HL +1 +2 +3 +4 +5 +6 +7 PSG アドレス データ アドレス データ アドレス データ アドレス データ (データ構造 8バイト)
0FE2H	CLDSTA	破壊 ——— 入力 ——— 出力 ———	モニタのコールドスタート
1003H	HOTSTA	破壊 ——— 入力 ——— 出力 ———	モニタのホットスタート
1143H	HEXCHL	破壊 AF, DE 入力 DE:16進数を表す文字のアドレス 出力 Acc:16進数, Cy=1:変換できない, DE:+2される	DEで示すアドレスの文字を16進数に変換してAccにセット
115EH	HEXCAL	破壊 AF, DE 入力 DE:16進数を表す2文字の文字列先頭アドレス(注2) 出力 Acc:16進数, Cy=1:変換できない, DE:+4される	DEで示すアドレスから2文字を16進数とみなしてAccにセット
11AFH	DUMP	破壊 AF, BC, HL 入力 HL:ダンプ開始アドレス, C:ダンプバイト数 出力 HL:HL+C	HLで示すアドレスからCバイトメモリダンプ
1202H	HLHEXP	破壊 AF, AF' 入力 HL:表示値 出力 ———	HLの値を4文字の16進数で表示
1207H	HEXPRT	破壊 AF, AF' 入力 Acc:表示値 出力 ———	Accの値を2文字の16進数で表示

注1, 異常コードAcc:00…OK, 01…BREAK, 02…CHECKSUM ERROR, VERIFY ERROR, 03…OUT OF TAPE, 04…WRITE PROTECT, 05…TAPE END, カセットキー, その他

注2, スペースはスキップ:16進数を表す文字が1文字ならDEは+1されるCy=1:DE不変化

BIOS ROMルーチン表(X1 turbo)

アドレス	ルーチン名	レ　ジ　ス　タ	説　　明
0000	IPLBOT	破壊　——— 入力　——— 出力　———	IPL コールドスタート
106C	WORKBS	破壊　HL, DE, BC 入力　——— 出力　———	BIOSのワークエリアをセット
1099	BIOSRS	破壊　HL, DE, BC, AF, HL', DE', BC', AF 入力　COLORF,CLSCHR,SCRMOD,WK1FD0 出力　———	I/Oイニシャライズ
10B4	PSGINT	破壊　HL, BC, AF 入力　——— 出力　———	PSGイニシャライズ
10D0	KVECIN	破壊　HL, AF, I 入力　——— 出力　———	割り込みによるKEY入力設定
10D6	KVEC00	破壊　AF 入力　L：$00 出力　———	割り込みを使用しないKEY入力モードの設定
10DF	SCRNSB	破壊　HL, DE, BC, AF, HL', DE', BC', AF' 入力　A：スクリーンモード番号 WIDTH0 A＝＊0H：WIDTH,25,0 A＝＊1H：WIDTH,12,0 A＝＊2H：WIDTH,20 A＝＊3H：WIDTH,10 A＝＊4H：WIDTH25,1 A＝＊5H：WIDTH,12,1 A＝0＊H：WIDTH,,, 0 A＝1＊H：WIDTH,,, 1 A＝2＊H：WIDTH,, 2 出力　———	スクリーンモードの設定
11D8	CR400S	破壊　HL, DE, BC, AF 入力　——— 出力　———	CRTCを400ライン用に設定
11E7	ROMASK	破壊　HL, DE, BC, AF, HL', DE', BC', AF' 入力　COLORF,CLSCHR, SCRMOD,WK1FD0 出力　———	ASKコマンド処理
1220	WITH80	破壊　HL, DE, BC, AF, HL', DE', BC', AF' 入力　COLORF,CLSCHR, SCRMOD, WK1FD0, GRAYMX,CURYHX 出力　———	WIDTH80の処理
1227	WITH40	破壊　HL, DE, BC, AF, HL', DE', BC', AF' 入力　COLORF,CLSCHR, SCRMOD, WK1FD0, GRAYMX,CURYHX 出力　———	WIDTH40の処理
12B9	CTRLD2	破壊　AF 入力　WIDTH0,CURYMX, WKTFD0,SCRNM3 出力　———	CONSOLEを標準にセットしてSCREEN0,0の処理
12DB	SCRNOT	破壊　AF 入力　A：ディスプレイモード 00：テキストページ0 グラフィックページ0 01：テキストページ1 グラフィックページ1 02：テキストページ0 グラフィックページ2 03：テキストページ1 グラフィックページ3 出力　———	SCREENの表示モードセット

アドレス	ルーチン名	レ ジ ス タ	説 明
1307	SCRNIN	破壊 AF 入力 A:ディスプレイモード 00:テキストページ0 グラフィックページ0 01:テキストページ1 グラフィックページ1 02:テキストページ0 グラフィックページ2 03:テキストページ1 グラフィックページ3 出力 ——	SCREENアクセスモードセット
134C	PALETI	破壊 D,BC,AF 入力 TPRIOF 出力 ——	すべてのパレットをイニシャライズ
1359	STPRIO	破壊 D,BC,AF 入力 BPRIOF,RPRIOF, GPRIOF,TPRIOF 出力 ——	プライオリティのセット
136C	PALETF	破壊 BC,AF 入力 —— 出力 ——	すべてのパレットを0にセット
1377	STCLST	破壊 HL,D,BC,AF 入力 COLORF,CLSCH 出力 ——	テキストV-RAMクリア
139A	STCLSG	破壊 AF,BC 入力 WK1FD0,SCRMOD 出力 ——	グラフィックRAMクリア
13E5	S49RES	破壊 DE,BC,AF 入力 —— 出力 ——	サブCPUの会話用バッファクリア
1408	IN49SB	破壊 FLG 入力 —— 出力 A:データ	サブCPUよりAレジスタにデータ入力
1413	OT49SB	破壊 FLG 入力 A:出力するデータ 出力 ——	サブCPUへAレジスタのデータ出力
143B	TAK49S	破壊 DE,B,AF 入力 A:コマンドNo DE~:データバッファ コマンド(16進数) D0~D7 タイマーセット(0~7)のセット(6バイト) D8~DF タイマー(0~7)からのリード(6バイト) E3:ゲームキーデータ読み込み(3バイト) E4:インタラプトによるKEY入力の割り込みベクタセット E5:タイマーをすべてクリア E6:KEYコードのリード(2バイト) E7:TV用コマンド書き込み E8:TV用コマンド読み込み E9:カセット用コマンド書き込み EA:カセット用コマンド読み込み EB:カセットのセンス EC:DATEセット(3バイト) ED:DATEリード(3バイト) EE:TIMEセット(3バイト) EF:TIMEリード(3バイト) 出力 ——	サブCPUとZ-80とのデータの入出力
1480	PALSET	破壊 AF,DE 入力 D:パレットコード(0~7) E:色 (0~7) 出力 ——	パレットのセット
14BF	INTCRT	破壊 DE,BC,AF,HL',DE',BC',AF' 入力 WIDTH0,GRAXMX,WK1FD0, GRAYMX,CURYMX 出力 ——	INIT "CRT:" の処理
1754	DEPRT	破壊 —— 入力 DE~:エンドコード=00H 出力 エンドコードのアドレス+1	文字列の表示

アドレス	ルーチン名	レ　ジ　ス　タ	説　　　明
1770	CR2	破壊　AF 入力　CURX,CURY 出力　———	改行されていなければ改行
1778	CR1	破壊　AF 入力　CURX,CURY 出力　———	改行(CR＋LF)
178F	SPPRT	破壊　——— 入力　CURX,CURY,COLORF 出力　———	スペースの表示
1791	ACCPRT	破壊　——— 入力　A:表示するASCIコード 　　　CURX,CURY 出力　———	1文字表示(コントロールコードは実行される)
179D	ACCDIS	破壊　——— 入力　A:表示するASCIIコード 　　　CURX,CURY,COLORF 出力　———	1文字表示(コントロールコードは実行されない)
18BC	ADRCA2	破壊　AF,BC 入力　L:X座標，H:Y座標 出力　HL:TEXT I/Oアドレス	X,Y座標よりテキストアドレスの計算
18E1	CTRLJB	破壊　AF,BC,DE,HL 入力　A:00H～1FH 出力　———	コントロールコードの処理
1B41	BEEP	破壊　AF,BC,DE,HL 入力　——— 出力　———	音を一瞬だけ出す
1D89	STRIGS	破壊　FLG,BC,HL 入力　A＝0:KEYコード入力 　　　A＝1:ジョイスティック1より入力 　　　A＝2:ジョイスティック2より入力 出力　A:＄20→ON 　　　A:＄00→OFF	ジョイステックトリガまたは，KEYコード入力
ID92	STICKS	破壊　FLG,BC,HL 入力　A＝0:KEYコード入力 　　　A＝1:ジョイスティック1より入力 　　　A＝2:ジョイスティック2より入力 出力　A:1～9に対するASCIIコードならば方向を示す	ジョイステック方向または，KEYコード入力
1DC2	BINPUT	破壊　AF 入力　DE:バッファ先頭アドレス 出力　DE～:入力されたデータ 　　　もしCY＝1ならブレイクでリターンした 　　　(SHIFT＋BREAK or CTRL-C)	INPUT
1DE4	INPUTF	破壊　AF 入力　DE:バッファ先頭アドレス 出力　BINPUT	LINPUT
1F16	BCUYST	破壊　AF,D 入力　H:CURY 出力　E:CURY,HL:FLG-ADR	Hレジスタに Y座標を与え，その行の始まっているY座標をDにかえす
1F25	ECUYST	破壊　AF,D 入力　H:CURY 出力　E:CURY,HL:FLG-ADR	Hレジスタに Y座標を与え，その行の最後＋1のY座標をDにかえす
1F8F	SCRGET	破壊　AF,AF',BC,DE,HL 入力　A:サイズ，E:X座標， 　　　D:Y座標， 　　　HL:データバッファアドレス 出力　———	SCRN＄
1FF0	INKEYS	破壊　FLG 入力　A＝0　:INKEY＄(0) 　　　A＝1　:INKEY＄(1) 　　　A＝2　:INKEY＄(2) 　　　A＝FFH:INKEY＄ 出力　A:KEYコード	INKEY＄の処理
20D5	BRKCKS	破壊　AF 入力　——— 出力　ZF＝1:押された	(SHIFT＋BREAK)または(CTRL-C)を押したかどうかのチェック

アドレス	ルーチン名	レジスタ	説明
20EB	KEYSNS	破壊 AF 入力 ——— 出力 ZF=0:データは有効	KEY入力チェック
274E	X1HPDS	破壊 AF, BC, DE, HL 入力 D:INKEY$2 KEYDAT+1, X1HELP, WIDTH0, X1MODE 出力 ———	XFERモードの表示
2A1B	FKYDSI	破壊 AF, BC, DE, HL 入力 ——— 出力 ———	ファンクションキーの表示
2A22	FKYDSS	破壊 AF, BC, DE, HL 入力 D:INKEY$(2) 出力 ———	ファンクションキーのモード表示
2A6B	EDLNDS	破壊 AF, BC, DE, HL 入力 WIDTH0, X1MODE, FKYDSF 出力 ———	XFER,ファンクションモード表示
32AD	CGSET	破壊 AF, BC, DE, HL 入力 DE:ASCIIコード HL～データバッファ 出力 CY=1:エラー	PCGのデータのセット
330D	CGREAD	破壊 AF, BC, DE 入力 DE:ASCIIコード HL～:データバッファ 出力 CY=1:エラー HL:次のデータバッファ	PCGのデータのメインRAMに転送
33C5	MONOP	破壊 AF, BC, DE, AF', BC', DE', HL', IX, IY 入力 ——— 出力 ———	モニタサブルーチン
3AF8	SUB	破壊 AF, BC, AF', BC', DE', HL' 入力 PRCSON:TYPE(2, 5, 8) HL～:データ1 DE～:データ2 出力 PRCSON:TYPE(2, 5, 8) HL～:答	減算 データ1－データ2
3AFB	ADD	破壊 AF, BC, AF', BC', DE', HL' 入力 PRCSON:TYPE(2, 5, 8) HL～データ1 DE～:データ2 出力 PRCSON:TYPE(2, 5, 8) HL～:答	加算 データ1＋データ2
3DBA	CMP	破壊 AF, B 入力 PRCSON:TYPE(2, 5, 8) HL～:データ1 DE～:データ2 出力 CYフラグとZフラグに比較結果	比較 データ1－データ2
3E01	MUL	破壊 AF, BC, AF', BC', DE', HL', IX, IY 入力 PRCSON:TYPE(2, 5, 8) HL～:データ1 DE～:データ2 出力 PRCSON:TYPE(2, 5, 8) HL～:答	乗算 データ1×データ2
40E3	DIV	破壊 AF, BC, AF', BC', DE', HL' 入力 PRCSON:TYPE(2, 5, 8) HL～:データ1 DE～:データ2 出力 PRCSON:TYPE(2, 5, 8) HL～:答	除算 データ1÷データ2
40E3	INTDVS	破壊 AF, BC 入力 DE:データ1 HL:データ2 出力 DE:商 HL:あまり	符号付整数の除算 データ1÷データ2
411D	INTDVN	破壊 AF, BC 入力 DE:データ1 HL:データ2 出力 DE:商, HL:あまり	符号なし整数の除算

アドレス	ルーチン名	レジスタ	説明
4122	INTDVV	破壊 AF,BC 入力 HL:データ1上位 DE:データ1下位 BC:データ2 出力 DE:商 HL:あまり	符号なし整数の除算 HLDE/BC:DE……HL
5296	CVDATS	破壊 AF,BC,HL 入力 ——— 出力 DE~:日付のASCIIコード文字列	日付の読み出し
5299	CVDATE	破壊 AF,BC 入力 HL:日付の内部表現 DE~:文字列バッファ 出力 DE~:日付のASCIIコード文字列	日付の内部表現をASCIIコード文字列に変換
52DF	CVDAYS	破壊 AF,BC,HL 入力 ——— 出力 DE~:曜日のASCIIコード文字列	曜日の読み出し
52E2	CVDAY	破壊 AF,BC 入力 HL~:曜日の内部表現 DE~:文字列バッファ 出力 DE~:曜日のASCIIコード文字列	曜日の内部表現をASCIIコード文字列に変換
52FB	CVTI$S	破壊 AF,BC,HL 入力 ——— 出力 DE~:時間のASCIIコード文字列	時間の読み出し(TIME$用)
5300	CVTIME	破壊 AF,BC 入力 HL~:時間の内部表現 DE~:文字列バッファ 出力 DE~:時間のASCIIコード文字列	時間の内部表現をASCIIコード文字列に変換(TIME$用)
5316	CVTIMS	破壊 AF,BC,HL,AF',BC',DE',HL',IX 入力 DE~:時間 PRCSON:TYPES 入力 DE~:TIMEバッファ	TIME変換用の時間の読み出し
532B	DATSTS	破壊 AF,BC,DE,HL 入力 DAYMES~:日付のASCIIコード表現 出力 ———	日付のセット
53A8	DAYSTS	破壊 AF,BC,DE,HL 入力 DAYMES~:曜日のASCIIコード表現 出力 ———	曜日のセット
53D7	TI$STS	破壊 AF,BC,DE,HL 入力 DAYMES:~時間のASCIIコード表現 出力 ———	時間のセット(TIME$用)
5418	TISTS	破壊 AF,BC,DE,HL,AF',BC',DE',HL 入力 DAYMES~:時間の内部表現 PRCSON:TYPE 出力 ———	時間のセット(TIME変数用)
5507	BOXFUL	破壊 AF,BC,DE,HL,BC',DE',HL',IX,IY 入力 LINEXS:先頭X座標 LINEXE:最終X座標 LINEYS:先頭Y座標 LINEYE:最終Y座標 PSMODE, CHRCOD COLORF, KSENFG 出力 ———	長方形を描きその中をぬりつぶす
5604	BOXSUB	破壊 AF,BC,DE,HL,BC',DE',HL',IX,IY 入力 LINEXS:先頭X座標 LINEXE:最終X座標 LINEYS:先頭Y座標 LINEYE:最終Y座標 PSMODE, CHRCOD COLORF, KSENFG 出力 ———	長方形を描く
569F	LINESB	破壊 AF,BC,DE,HL,BC',DE',HL',IX,IY 入力 LINEXS:先頭X座標 LINEYS:先頭Y座標 LINEXE:最終X座標 LINEYE:最終Y座標 PSMODE, CHRCOD COLORF, KSENFG 出力 ———	直線を引く

アドレス	ルーチン名	レ ジ ス タ	説 明
578D	ELHPUT	破壊 AF 入力 BC:グラフィックアドレス E:青のデータ L:赤のデータ H:緑のデータ 出力 ———	PUT
57AA	ELIHGET	破壊 AF, E, HL 入力 BC:グラフィックアドレス E:青のデータ L:赤のデータ H:緑のデータ 出力 ———	GET
57F1	PSETSB	破壊 AF, BC, DE, HL 入力 PSETX:X座標 PSETY:Y座標 GCOLOR:色データ 出力 ———	PSET
580C	RESETS	破壊 AF, BC, DE, HL 入力 PSETX:X座標 PSETY:Y座標 GCOLOR:色データ 出力 ———	RESET
58BD	POINTS	破壊 AF, BC, DE, HL 入力 DE:X座標 HL:Y座標 SCRNM2スクリーンモード (0:カラー, 1.2.3:モノクロ) 出力 A:その座標のデータ(0−7) CY:1→WINDOW OVER	Aレジスタ←〔DE(X座標), HL(Y座標)〕
5907	GRAADR	破壊 AF, BC, DE, HL 入力 DE:X座標 HL:Y座標 SCRNM2スクリーンモード (0:カラー, 1.2.3:モノクロ) 出力 HL:アドレス	グラフィックアドレス計算とWINDOWのチェック
590F	GRAAD2	破壊 AF, BC, DE, HL 入力 DE:X座標 HL:Y座標 SCRNM2スクリーンモード (0:カラー, 1.2.3:モノクロ) 出力 HL:アドレス	グラフィックアドレス計算
59A8	UPADR	破壊 AF 入力 BC:グラフィックアドレス WK1FD0, WIDTH0, SCRNW3 出力 BC:グラフィックアドレス	1ドット上のグラフィックアドレス計算
59FC	DWADR	破壊 AF 入力 BC:グラフィックアドレス WK1FD0, WIDTH0, SCRNW3 出力 BC:グラフィックアドレス	1ドット下のグラフィックアドレス計算
5A4D	CLSGRA	破壊 AF, BC, DE, HL 入力 CLSMOD 出力 ———	G-RAMのクリア
5AD8	WINDOI	破壊 AF, BC, DE, HL 入力 ——— 出力 ———	WINDOWを最大にする
5AEA	WINDST	破壊 AF, BC, DE, HL, AF', BC', DE', HL' 入力 HL:X座標の最小値 DE:Y座標の最小値 出力 GCURXS, GCURYS, GUURXE, GCURYE CLSXLN, CLSYLN, CLSECD, CLSFCD, SCRNXS, SCRNXE, SCRNYS, SCRNYE, WIBIXE, WIBYXE, WIBIXS, WIBYXE	WINDOWをセットする
5B99	TILCOL	破壊 AF, BC, DE, HL 入力 GCOLOR, TILBUF 出力 TILBUF	カラーパターンセット

アドレス	ルーチン名	レ ジ ス タ	説 明
5EA1	HPAINT	破壊 AF, BC, DE, HL, BC', DE' 入力 PAINTX, PAINTY, GCOLOR 出力 ――	ペイント
61A5	TILSET	破壊 AF, HL 入力 A:バッファ番号(0〜7) 出力 ――	タイルバッファにタイルパターンのセット
623D	PATSUB	破壊 AF, BC, DE, HL, BC', DE' 入力 GCURX:X座標 GCURY:Y座標 PATUDD:パターンのY方向の長さ DE:データアドレス A:データの長さ 出力 ――	PATTERN処理
630B	POLYSB	破壊 ―― 入力 SINSX, SINSY, SINRX, SINRY, GCOLOR, SIND, SINSTA, SINEND 出力 ――	多角形を描く
656E	TEMPSB	破壊 AF, BC, DE, HL 入力 DE:TEMPO(30〜7500) 出力 ――	テンポのセット
65AC	MUSICS	破壊 AF, BC, DE, HL 入力 DE:データアドレス(データの最後は00H) HL:インタラプトバッファ A:モード(0:エンドウェイト, 1:スタートウェイト) 出力 ――	MUSIC
65F2	MUBFST	破壊 AF, C 入力 DE:MUSICデータアドレス HL:インタラプトデータアドレス 出力 DE:次のMUSICデータアドレス HL:次のデータアドレス	MUSICデータをインタラプト用のデータに変換
66A3	MUSICI	破壊 AF, HL 入力 CHAADR, CHBADR, CHCADR 出力 ――	インタラプトでMUSICを出す
67A7	HCOPSS	破壊 AF, BC, DE, HL 入力 A:モード A=FFH:TEXT A=0:G-RAMすべて A=1:G1 A=2:G2 A=3:G3 A=4:TEXTとG-RAMすべて SCRNM3, HCYMIN, HCXMAX, HCYMAX, LDCRCD, CURYMX, SCRN00, INIAD 出力 ――	HCOPY
6AD6	BITDEP	破壊 AF 入力 LDTBUF, LPOSB 出力 ――	プリンタにビットパターンの出力
6BD3	SYMBSB	破壊 AF, BC, DE, HL, AF', BC', DE', HL', IX, IY 入力 SINSX, SINYS, GETADR, PSMODE 出力 ――	SYMBOL
6D3F	SIOCTC	破壊 AF, BC, DE, HL 入力 ―― 出力 ――	CTCとSIOのイニシャライズ
6DA5	RSINIT	破壊 AF, BC, DE, HL 入力 H:CTC1のデータ L:SIOA-R4 D:SIOA-R5 E:SIOA-R3 出力 ――	SIO-Aをすべてのモードセット
6E59	RXINP	破壊 AF 入力 ―― 出力 A:入力したデータ	RS-232Cよりデータ入力
6E83	RXSNS	破壊 AF, HL 入力 ―― 出力 ZF:1→データなし ZF:0→データ入力OK	RS-232C入力センス

アドレス	ルーチン名	レ　ジ　ス　タ	説　　明
6EBA	TXOUT	破壊　FLG 入力　A:出力するデータ 出力　——	RS-232Cへデータ出力
6EA7	TXSNS	破壊　AF,BC 入力　—— 出力　ZF:1→出力不可 　　　ZF:0→出力OK	RS-232C出力センス
6EAF	MOUSE0	破壊　AF,BC 入力　—— 出力　——	MOUSE0,CTC0をインタラプトモードで使用しない
6EC0	MOUSE1	破壊　AF,BC,DE,HL 入力　HL:X座標 　　　DE:Y座標 出力　——	マウスポジションセット
7020	SAVE1	破壊　AF,BC 入力　HL:先頭アドレス 　　　DE:サイズ(=20H) 出力　A=0:OK 　　　A=1:ブレイク 　　　A=3:テープがセットされていない 　　　A=4:消去防止のツメが折れている 　　　A=5:テープエンド	ファイルネームをカセットに記録
7024	SAVE2	破壊　AF,BC 入力　HL:先頭アドレス 　　　DE:サイズ 出力　A=0:OK 　　　A=1:ブレイク 　　　A=3:テープがセットされていない 　　　A=4:消去防止のツメが折れている 　　　A=5:テープエンド	データをカセットに記録
7047	LOAD1	破壊　AF,BC 入力　HL:先頭アドレス 　　　DE:サイズ(=20H) 出力　A=0:OK 　　　A=1:ブレイク 　　　A=2:チェックサムエラー 　　　A=3:テープがセットされていない 　　　A=5:テープエンド	ファイルネームをカセットから読み込む
704B	LOAD2	破壊　AF,BC 入力　HL:先頭アドレス 　　　DE:サイズ 出力　A=0:OK 　　　A=1:ブレイク 　　　A=2:チェックサムエラー 　　　A=3:テープがセットされていない 　　　A=5:テープエンド	カセットからメモリへロード
705C	VERFY2	破壊　AF,BC 入力　HL:先頭アドレス 　　　DE:サイズ 出力　A=0:OK 　　　A=1:ブレイク 　　　A=2:チェックサムエラー 　　　A=3:テープがセットされていない 　　　A=5:テープエンド	カセットとメモリの比較
72C3	CMTCOM	破壊　—— 入力　A= 0 :EJECTする 　　　A= 1 :STOP 　　　A= 2 :PLAY 　　　A= 3 :早送り 　　　A= 4 :巻きもどし 　　　A= 5 :APSS(FF) 　　　A= 6 :APSS(REW) 　　　A=10:REC 出力　——	カセットレコーダのコントロール
72CD	CMTSNS	破壊　AF 入力　—— 出力　Aのbit0=0:テープエンド,Aのbit1=0:カセットなし 　　　Aのbit2=0:消去防止ツメなし	カセットレコーダの状態を読み出し

アドレス	ルーチン名	レジスタ	説明
739D	FDCRED	破壊 AF,BC,DE,HL,AF' 入力 HL:データ先頭アドレス DE:レコードナンバー A:レコードサイズ FDCNO(デバイス番号),UNITNO(ドライブ番号) 出力 ———	デバイスから読み出し(G-RAM,外部RAM,3インチ,5インチ,8インチ,ハードディスク)
73AA	FDCWRT	破壊 AF,BC,DE,HL,AF' 入力 HL:データ先頭アドレス DE:レコードナンバー A:レコードサイズ FDCNO(デバイス番号),UNITNO(ドライブ番号) 出力 ———	デバイスへ書き込み(G-RAM,外部RAM,3インチ,5インチ,8インチ,ハードディスク)
73B7	FDCVFY	破壊 AF,BC,DE,HL,AF' 入力 HL:データ先頭アドレス DE:レコードナンバー A:レコードサイズ FDCNO(デバイス番号),UNITNO(ドライブ番号) 出力 ———	デバイスの比較(G-RAM,外部RAM,3インチ,5インチ,8インチ,ハードディスク)
76CA	DSKRED	破壊 AF,BC,DE,HL,AF' 入力 HL:データ先頭アドレス D:先頭トラック(0〜79) E:先頭セクタ(1〜16) A:セクタの長さ 出力 ———	3インチ,5インチディスク読み込み (DMAを使用しない)
76D5	DSKWRT	破壊 AF,BC,DE,HL,AF' 入力 HL:データ先頭アドレス D:先頭トラック(0〜79) E:先頭セクタ(1〜16) A:セクタの長さ 出力 ———	3インチ,5インチディスク書き込み (DMAを使用しない)
76E0	DSKVFY	破壊 AF,BC,DE,HL,AF' 入力 HL:データ先頭アドレス D:先頭トラック(0〜79) E:先頭セクタ(1〜16) A:セクタの長さ 出力 ———	3インチ,5インチディスクのベリファイ 3インチ,5インチディスクのベリファイ
7792	MOTOFS	破壊 AF,BC 入力 ——— 出力 ———	3インチ,5インチディスクモータストップ
7797	MOTOFF	破壊 AF,BC 入力 ——— 出力 ———	3インチ,5インチディスクモータOFF
78D9	HDINIT	破壊 AF,BC,DE,HL,IY 入力 A:ドライブ番号(0〜15) 出力 ———	ハードディスクのイニシャライズ
78E2	HDOFFS	破壊 AF,BC,DE,HL,IY 入力 A:ドライブ番号(0〜15) 出力 ———	ハードディスクのOFF

BIOSワークエリア表

F800	INTTAB	F928	MTOFIO	FAED	KEYDAT
F83C	BIOSER	F929	HDDMAS	FAEE	KEYDA2
F843	KEYRAM	F96E	DSKWKS	FAEF	COU1MS
F847	INTSUB	F994	DMADSK	FAF0	BRKBUF
F876	X1MODB	F9A4	MEMEMM	FAF1	ONKYBF
F877	OPTKEY	F9BC	LDABCS	FAF2	RSINTF
F879	HELPKY	F9C0	LDABCT	FAF3	RSERRF
F87A	COPYKY	F9CC	LDBCAS	FAF4	INTFLG
F87B	GRAXMX	F9D5	LDBCAT	FAFE	POINT1
F87D	GRAYMX	F9DC	CPABCS	FAFF	POINT2
F87F	WIDTH0	F9E3	CPABCT	FB00	INBUF
F880	CURYMX	F9EC	JPRET	FB40	POINT3
F881	VRMGAI	F9F7	LDCHL	FB41	INPBUF
F882	PRTGAI	F9FE	BKLDIR	FB6A	INIADR
F883	LPCRCD	FA06	BKLDDR	FB6C	INIADW
F884	LPPGCD	FA0E	DEHLCK	FB6E	KANJIF
F885	LPTGIC	FA25	HLDECK	FB6F	KBUFBW
F88A	LPTLSC	FA39	SETRES	FB70	CSIZEF
F88F	LPTBTC	FA3D	SETMD	FB71	LPOSB
F894	LPTLNC	FA40	RESMD	FB72	LPOSA
F899	LPTGOC	FA46	X1MDCL	FB73	LPOSK
F89E	LPTKIC	FA47	X1SLCL	FB74	FILOUT
F8A2	DOTSPC	FA48	CLICKM	FB75	ESCFLG
F8A7	KLFTDT	FA50	INSPRT	FB76	ESCPRF
F8A8	KRGTDT	FA51	POWERF	FB77	CTRLAF
F8A9	LPTKOC	FA52	SEED	FB78	KEYFLG
F8AD	LPACHN	FA54	MEMMAX	FB79	GRACOD
F8AE	PRTDLY	FA56	HCXMIN	FB7A	ROMFLG
F8AF	LPTABL	FA57	HCXMAX	FB7B	CTRLMD
F8B0	VRMPRB	FA58	HCYMIN	FB7C	SIOBR5
F8B2	RLARRA	FA59	HCYMAX	FB7D	CHRAND
F8B7	OPENF9	FA5A	MOUSX1	FB7E	MONSP
F8BA	OPENF8	FA5C	MOUSY1	FB80	CHAADR
F8BD	OPENF7	FA5E	MOUSX2	FB82	MUAADR
F8C0	FINDF7	FA60	MOUSY2	FB87	MUACOU
F8C3	NEXTF7	FA62	MOUSXD	FB88	CHBADR
F8C6	BACKF7	FA63	MOUSYD	FB8A	MUBADR
F8C9	X1CLF7	FA64	TABBUF	FB8F	MUBCOU
F8CC	NEXTJS	FAB4	FD5DRT	FB90	CHAADR
F8CF	LINLIM	FAB8	FD8DRT	FB92	MUAADR
F8D0	COLORF	FAB9	DMAIOF	FB97	MUACOU
F8D1	CLSCHR	FABA	FUNADR	FB98	MOUSEX
F8D2	BPRIOF	FABC	FKYDSF	FB9A	MOUSEY
F8D3	RPRIOF	FABD	DIRIMG	FB9C	MOUSEF
F8D4	GPRIOF	FADD	FDCNO	FB9D	MOUPNT
F8D5	TPRIOF	FADE	UNITND	FB9E	MOUDAT
F8D6	WK1FD0	FADF	CURX	FBA1	MS1OFX
F8D7	SCRMOD	FAE0	CURY	FBA3	MS1OFY
F8D8	SECMIN	FAE1	COPYXY	FBA5	MS1ONX
F8D9	SECMAX	FAE3	CURYST	FBA7	MS1ONY
F8DA	PRCSON	FAE4	CURYED	FBA9	MS2OFX
F8DB	REPTF1	FAE5	CURXST	FBAB	MS2OFY
F8DC	TMPEND	FAE6	CURXED	FBAD	MS2ONX
F8DE	BITDES	FAE7	LPOSST	FBAF	MS2ONY
F8E1	HCOPYS	FAE8	LPOSLN	FBB1	RSSTCT
F8E4	CPSM23	FAE9	LPPAGE	FBB2	RSPNT1
F8E7	KEYSNN	FAEA	LPPGST	FBB3	RSPNT2
F8EA	SCRRAM	FAEB	LPPGLN	FBB4	RSBUF
F8ED	RAMCR1	FAEC	CLICKF	FBF4	SCRN00

FBF5	SCRN01
FBF6	SCRNM2
FBF7	SCRNM3
FBF8	SCRNM4
FBF9	KSENFG
FBFA	INTMUF
FBFB	VFLAG
FBFC	GCURXS
FBFE	GCURYS
FC00	GCURXE
FC02	GCURYE
FC04	WIBYXS
FC05	WIBIXS
FC06	WIBYXE
FC07	WIBIXE
FC08	CLSECD
FC09	CLSFCD
FC0A	CLSXST
FC0B	CLSXLN
FC0C	CLSYLN
FC0E	SCRNXS
FC10	SCRNXE
FC12	SCRNYS
FC14	SCRNYE
FC16	GCOLOR
FC17	GETXS
	LINEXS
	PSETX
FC19	GETYS
	LINEYS
	PSETY
FC1B	GETXE
	LINEXE
FC1D	GETYE
	LINEYE
FC1F	GCURX
FC21	GCURY
FC23	SCRNTD
FC3D	SCRNT1
FC57	SCRNTC
FC59	DSKTRK
FC5D	DSKSTK
FC61	DKIOSW
FC62	COMLIN
FC63	DSKERR
FC64	SCRLAD
FC66	SUMDT
FC68	TIMBUF
FC6D	LPTBUF
FCF9	HIRAFL
FCFA	KANBUF
FD36	ONEBUF
FD38	ONESTA
FD3A	ONEEND
FD3C	HENBUF
FD65	HENASC
FD70	X1HELP
FD71	X1FUNC
FD72	X1MODE
FD73	X1POS
FD74	X1ESCF
FD75	RMAASC
FD79	RMAKAN
FD7D	COPYMD
FD7F	HCOPYB
FD9F	DAYMEB
FDA7	DATEBF
FDA8	DAYBF
FDAA	TIMEBF
FDAD	NESTAK
FDAF	HDBORD
FDB1	CMDTBL
FDB2	HDDRV
FDB3	HDREC
FDB5	HDLEN
FDB7	HDSPCB
	BCOUNT
FDB8	CYFLG
FDB9	ZFAC
FDC1	ZFAC1
FDC9	ZFAC2
FDD1	DGITCO
FDD2	DGITFG
FDD3	EXPFLG
FDD4	PRODFL
FDD5	DGBFM3
FDDC	DGBFM1
FDDD	DGBF00
FDE5	DGBF08
FDE7	DGBF10
FDE8	DGBF11
FDE9	DGBF12
FDED	DGBF16
FDEE	DGBF17
FDD5	CLIPX1
FDD7	CLIPY1
FDD9	CLIPX2
FDDB	CLIPY2
FDDD	WINDX1
FDDF	WINDY1
FDE1	WINDX2
FDE3	WINDY2
FE00	FATBUF
FE0F	SNFAC0
FE11	SNFAC1
FE13	SNFAC2
FE15	SNFAC3
FE17	SNFAC4
FE19	SNFAC5
FE1B	EXPSIN
FE1C	EXPOFF
FE1D	EXPHBT
FE1E	LOGEXP
FE1F	SINSGN
FE20	TILBUF
FE38	BAKBUF
FE48	BKCOLR
FE50	BKCLLN
FE51	CHRCOD
FE52	CLSMOD
FE53	PUTMOD
	PSMODE
FE54	LINPAT
FE56	PATUDD
FE58	FOLCIR
FE59	PAINTX
	SINSX
FE5B	PAINTY
	SINSY
FE5D	SINRX
FE5F	SINRY
FE61	SIND
FE63	GETADR
	SINSTA
FE65	ARYEDA
	SINEND
FE67	XMULHI
FE68	XMULLO
FE6A	YMULHI
FE6B	YMULLO
FE6D	SINXAD
FE6F	SINYAD
FE71	ENTPY
FE73	LX
FE74	LP
FE75	LA
FE77	RX
FE79	RA
FE7B	OLX
FE7D	OLA
FE7F	ORX
FE81	ORA
FE83	OOLX
FE85	OOLA
FE87	OORX
FE89	OORA
FE8B	CHKX
FE8D	STFLAG
FE8E	LKFLAG
FE8F	STKTOP
FE91	STKBTM
FE93	NXTPSH
FE95	NXTPOP
FE97	TILLBF
FE9A	XINC0
FE9C	XINC1
FE9E	YINC2
FEA0	YINC3
FEA2	XMOD
FEA4	YMOD
FF00	DIRBUF
	KEYBUF
FF87	IPLDRV
FF8C	DSKTYP

資　料

■MB8877A フロッピディスクコントローラ

MB8877Aは富士通社製フロッピディスク・フォーマッタ/コントローラ専用LSIであり，ディスクに対してデータの書き込み，読み出し，ならびにディスクドライブのメカニカル部分の駆動や制御信号の検出を行います。

MB8877Aフロッピディスクコントローラの端子配列図およびブロックダイアグラムを示し，各端子の機能を図にまとめます。

レジスタ

1.コマンドレジスタ(CR)

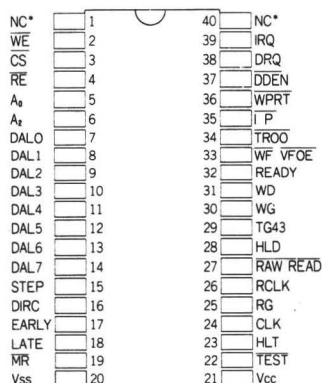
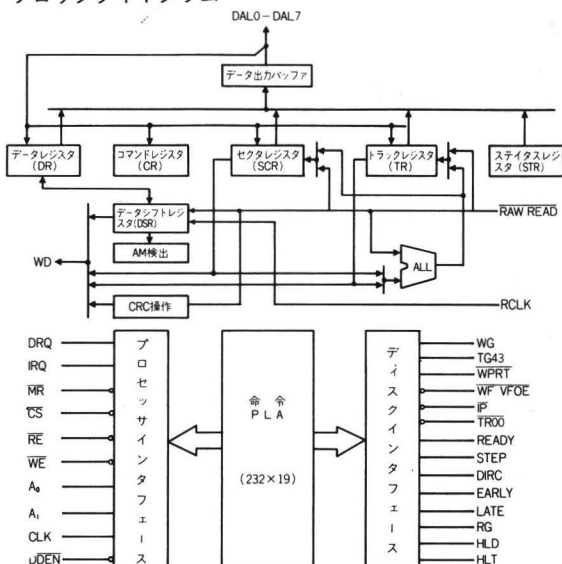
書き込み専用の8ビットレジスタです。このレジスタにはFDCに実行させる動作に応じたコマンドを，プロセッサ側から書き込みます。コマンドの書き込み動作は，フォースインタラプトコマンドを除き，FDCが前コマンド動作の終了後でなければなりません。

2.ステイタスレジスタ(STR)

読み出し専用の8ビットレジスタです。このレジスタは，FDCの内部状態，コマンド実行における処理状態，ディスクドライバの状態を示します。各ビットの意味は，実行中および実行完了したコマンドにより変化します。

3.データレジスタ(DR)

書き込み，読み出し可能の8ビットレジスタです。ディスクリード時には，ディスクか

端子配列図(TOP VIEW/MB8877A)

NC*：No internal connection

ら読み出されたデータがロードされ，ディスクライト時には，このレジスタに書き込まれたデータがディスクに書き込まれます。

シーク動作時には，目的のトラック番号を書き込みます。

4.データシフトレジスタ(DSR)

このレジスタは，外部からアクセスすることはできない8ビットのシフトレジスタです。

ディスクライト時には，DRからDSRへ並列転送されたデータが，ライトデータとしてFMもしくはMFMで変調され，WD端子から直列に出力されます。

ディスクリード時には，$\overline{\text{RAW READ}}$端子から入力するデータが直列にDSRに入り，復調されて1バイト単位でDRへ並列転送されます。

5.トラックレジスタ(TR)

書き込み，読み出し可能な8ビットレジスタです。このレジスタは，マスタリセット($\overline{\text{ML}}$＝“L”から“H”の立ち上がり)によって$(FF)_H$から順次減少し，$\overline{\text{TR00}}$が“L”レベルになったときに$(00)_H$となります。

このレジスタは通常ディスクのリード/ライト用ヘッドのあるトラックの番号がセットされます。FDCはトラックにより，この値を更新することも更新しないことも可能です。

リードデータ，ライトデータコマンドでは，この内容とディスクから読み出されたIDフィールドのトラック番号を比較して一致したトラックに対しリード/ライトを行います。

6.セクタレジスタ(SCR)

書き込み，読み出し可能な8ビットシフトレジスタです。リードデータ，ライトデータコマンドでは，このレジスタとディスクから読み出されたIDフィールドのセクタ番号を比較し，一致したセクタに対してリード/ライトを行います。

リードアドレスコマンドでは，IDフィールドのトラック番号が保持されます。

プロセッサインタフェース

プロセッサインタフェースは，FDC内部のレジスタと，プロセッサのデータ転送を行うためのインタフェースです。プロセッサはFDCに必要に応じたデータ，コマンドなどをこのインタフェースを介して，内部レジスタと転送を行います。

このインタフェースの信号線には，データ線と，レジスタ選択線，リード線，ライト線，チップセレクト線があります。プロセッサは，チップセレクト線，レジスタ選択線，データ線に所要のレベルを与えて，リード線もしくはライト線を制御して，データ転送を行います。

レジスタ選択

$\overline{CS}$	A_1 A_0	$\overline{RE}$=“L”	$\overline{WE}$=“L”
“H”	X　X	NON SELECT	DAL0-DAL7はHiインピーダンス
“L”	“L” “L”	ステイタスレジスタ	コマンドレジスタ
“L”	“L” “H”	トラックレジスタ	トラックレジスタ
“L”	“H” “L”	セクタレジスタ	セクタレジスタ
“L”	“H” “H”	データレジスタ	データレジスタ

内部レジスタの選択を右表に示します。

また，コマンド動作終了の通知の機能を持つ割り込み要求線(IRQ)およびディスクとのデータ転送におけるデータ読み出し，書き込み要求線(DRQ)があります。これによりプロセッサのプログラムの軽減，DMA転送のタイミング記号作成の軽減を可能にしています。

フロッピディスクインタフェース

FDCは，フロッピディスクインタフェースとして，ディスクのリード/ライト用ヘッドの駆動用信号，ディスクからのデータ読み取り用端子，ディスクへの書き込みデータ信号，フロッピディスクの状態検出用端子があります。

FDCは，プロセッサインタフェースよりコマンドレジスタに書き込まれたコマンドに応じて，フロッピディスクインタフェースを通じて，フロッピディスクを制御します。

コマンド説明

FDCには11種類のコマンドがあり，各コマンドは細部にわたる動作を決めるフラグを持ち，多くの動作モードを実現しています。

これらのコマンドは，4種に大別され，これをタイプⅠからタイプⅣとします。

各コマンドはタイプⅣのコマンドを除き，前のコマンドが終了した後(ステイタスのBUSY=0のとき)でなければ書き込んではなりません。

コマンド表

タイプ	コマンド名称	動　　作
Ⅰ	リストア	トラック0へ，ヘッドを移動する
	シーク	所定のトラックへ，ヘッドを移動する
	ステップ	ヘッドを1トラック移動する
	ステップイン	ヘッドを1トラック内側へ移動する
	ステップアウト	ヘッドを1トラック外側へ移動する
Ⅱ	リードデータ	ディスクのデータ(データフィールド)を読む
	ライトデータ	ディスク(データフィールド)へデータを書込む
Ⅲ	リードアドレス	ディスクのIDフィールドを読む
	リードトラック	ディスクの1トラック分の全データを読む
	ライトトラック	ディスクへ1トラック分の全データを書き込む
Ⅳ	フォースインタラプト	割り込みを発生させる

タイプⅠコマンド

タイプⅠコマンドは，ヘッドの移動とトラックの照合フラグを有します。

タイプⅠコマンド表

コマンド名称	コマンドレジスタビット (MSB) 7 6 5 4 3 2 1 0 (LSB)
リストア	0 0 0 0 h V r_1 r_0
シーク	0 0 0 1 h V r_1 r_0
ステップ	0 0 1 u h V r_1 r_0
ステップイン	0 1 0 u h V r_1 r_0
ステップアウト	0 1 1 u h V r_1 r_0

タイプⅠコマンドは，ステップレートフラグ($r_1 r_0$)，トラック照合フラグ(V)，ヘッドロードフラグ(h)，トラックレジスタ更新フラグ(u)を持っています。

ステップレートフラグ($r_1 r_0$)は，ステップパルス出力(STEP)の出力間隔(ステップレート)を指定します。ステップレートは，この$r_1 r_0$と，FDCに与えられるクロック(CLK)の周波数，$\overline{\mathrm{TEST}}$端子の状態によって右表のようになります。

ステップレートの変化

$\overline{\mathrm{TEST}}$		"H"もしくは開放		"L"	
$r_1 r_0$ ＼ CLK		2 MHz	1 MHz	2 MHz	1 MHz
0	0	3 ms	6 ms	Approx 200 μs	Approx 400 μs
0	1	6 ms	12ms		
1	0	10ms	20ms		
1	1	15ms	30ms		

ヘッドロードフラグ(h)は，コマンド実行開始時にヘッドをロードするか，ヘッドをメディアから離すかを指示します。

h=1：コマンド実行開始時にヘッドロードします。

h=0：コマンド実行開始時にヘッドを離します。

トラックレジスタ更新フラグは，ヘッド移動に際し，トラックレジスタを更新することを指示します。

u=1：トラックレジスタを更新します。

u=0：トラックレジスタを更新しません。

トラック照合フラグ(V)は，ヘッド移動後，ディスクのトラック番号とトラックレジス

タの照合を行うかを指示します。

V＝1：トラックの照合を行います。

V＝0：トラックの照合を行いません。

タイプIIコマンド

タイプIIコマンドは，ディスクのデータフィールドに対する書き込みと読み出し動作をします。

対象となるセクタ番号とトラック番号はそれぞれ，セクタレジスタ，トラックレジスタに用意されていなければなりません。ディスクとのデータ転送はデータレジスタを介して行います。

タイプIIコマンドは，対象セクタを探索し，そのセクタに対してリード/ライトを行います。対象セクタ検索はIDフィールドを読み，トラック番号とトラックレジスタ，セクタ番号とセクタレジスタを比較し，IDフィールドのCRCをチェックして行います。オプションとして，サイド番号をチェックすることも可能です。

タイプIIコマンド表

コマンド名称	コマンドレジスタビット (MSB) 7 6 5 4 3 2 1 0 (LSB)
リードデータ	1 0 0 m S E C 0
ライトデータ	1 0 1 m S E C a_0

1セクタに対して，リード/ライトの必要なバイト数はIDフィールドを読んだときのセクタ長指定バイトによってFDC内で決定されます。セクタ長指定バイトの内容とセクタ長の関係を次に示します。

タイプIIコマンドは，マルチレコードフラグ(m)，ディレイフラグ(E)，サイドフラグ(S)，比較フラグ(C)を持っています。また，ライトデータコマンドでは，アドレスマークフラグ(a_0)を持っています。

セクター長指定バイト

セクタ長指定バイト	セクタのデータ量
$(00)_H$	128バイト
$(01)_H$	256バイト
$(02)_H$	512バイト
$(03)_H$	1024バイト

マルチレコードフラグ(m)：

m＝1：連続セクタ(セクタ番号が増加する方向)でリード/ライトを行います。

m＝0：単一セクタでリード/ライトを行います。

デレイフラグ(E)：

E＝1:HLDを“H”としたのち，15ms待ちHLTをサンプリングします。

E＝0:HLDを“H”としたのち，ただちにHLTをサンプリングします。

比較フラグ(C)：

C＝1：サイド番号の比較を行います。

C＝0：サイド番号の比較を行いません。

サイドフラグ(S)：

S＝1：サイド番号のLSBが1のとき一致したものとみなします。

S＝0：サイド番号のLSBが0のとき一致したものとみなします。

(このフラグは，Cフラグが1のときのみ有効)

アドレスマークフラグ(a_0)：

a_0＝0：データアドレスマークに$(FB)_H$ (Data Mark)を書きます。

$a_0=1$：データアドレスマークに(F8)H (Deleted Data Mark)を書きます。

タイプⅢコマンド

タイプⅢコマンドは，リードアドレス，リードトラック，ライトトラックコマンドがあります。これらのコマンドは，現在ヘッドのあるトラックに対して処理が行われます。

リードアドレスコマンドは最初に出合ったIDフィールドの内容を読みだします。IDフィールドは6バイトであり，その順序と内容は，

1．トラックアドレス(トラック番号)
2．サイドナンバー　(ディスク面番号)
3．セクタアドレス　(セクタ番号)
4．セクタ長　　　　(セクタ長指定バイト)
5．CRC1
6．CRC2　(本IDフイールドの CRC 2バイト)です。

タイプⅢ コマンド表

コマンド名称	コマンドレジスタビット (MSB) 7 6 5 4 3 2 1 0 (LSB)
リードアドレス	1 1 0 0 0 E 0 0
リードトラック	1 1 1 0 0 E 0 0
ライトトラック	1 1 1 1 0 E 0 0

リードトラックコマンドは，$\overline{\mathrm{IP}}$(インデックスパレス)入力を検出した後ディスク内の全データを読み出します。データの同期は，インデックスマーク，IDアドレスマーク，データアドレスマークで行われます。

ライトトラックコマンドでは，最初の1バイトがデータレジスタに書き込まれてから，$\overline{\mathrm{IP}}$を検出し，第1バイトを書き込みます。これに続いてディスクに書き込むべきデータを1バイトずつデータレジスタに転送しなければなりません。もし，1バイトのデータがディスクに書き込まれてもデータレジスタに書き込み用の転送を怠ると，$(00)_H$が書き込まれたものとし，ディスクへ書き込みます。この動作を次の$\overline{\mathrm{IP}}$が検出されるまで繰り返します。

また，このコマンドではディスクイニシャライズのために，$(F5)_H$～$(FE)_H$がデータレジスタに書き込まれると，特別の処理を行います。

タイプⅣコマンド

タイプⅣコマンドは，フォースインタラプトコマンドです。このコマンドはいつもコマンドレジスタに書き込むことにより実行されます（他のコマンドは，実行中のコマンドがあるときに，新たにコマンドを発すると，その動作は保障されません)。

タイプⅣコマンド表

コマンド名称	コマンドレジスタビット (MSB) 7 6 5 4 3 2 1 0 (LSB)
フォース インタラプト	1 1 0 1 I_3 I_2 I_1 I_0

フォースインタラプトコマンド

I_i	IRQ発生条件
$I_0=1$	REDY入力の立ち上がりでIRQ発生(IRQ="H")
$I_1=1$	REDY入力の立ち下りでIRQ発生
$I_2=1$	各インデックスパルス（$\overline{\mathrm{IP}}$）でIRQ発生
$I_3=1$	無条件でただちにIRQ発生

フォースインタラプトが書き込まれると，指定に応じた条件でIRQが“H”レベルになります。もし，このコマンドが書き込まれたとき，他の実行中のコマンドがあると，実行中のコマンドは打ち切られ，フォースインタラプトコマンドの実行をします。

ステイタス

FDCは，コマンドの実行結果，実行中の状態を示すためにステイタスレジスタがありま

す。コマンドが実行されると，所要のビットがプリセットされ，コマンド実行中に更新され，コマンド実行終了直前に内容が確定します。各ビットの示す意味は，実行中のコマンド，実行終了したコマンドにより変化します。

ステイタスレジスタ

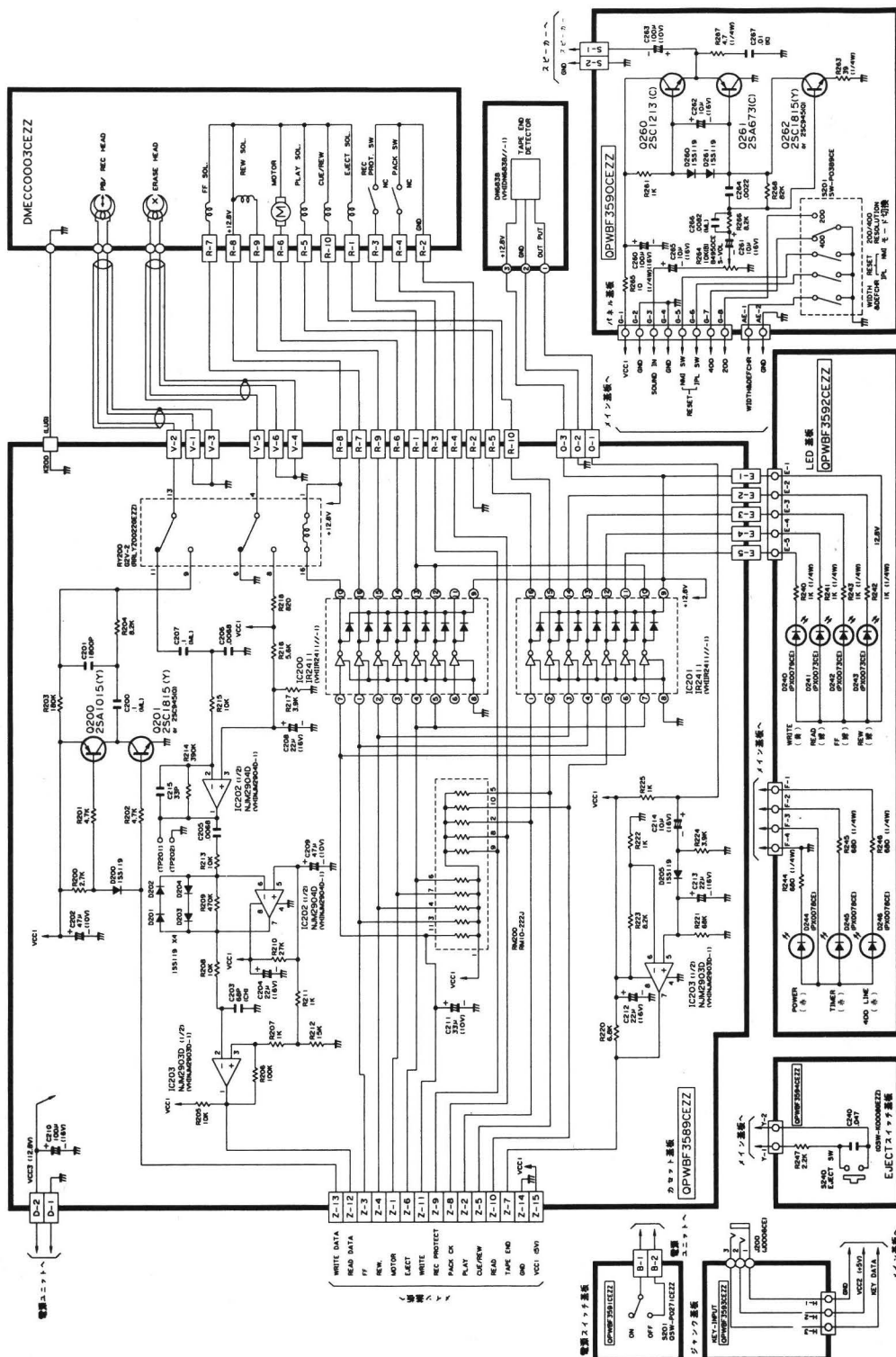
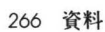
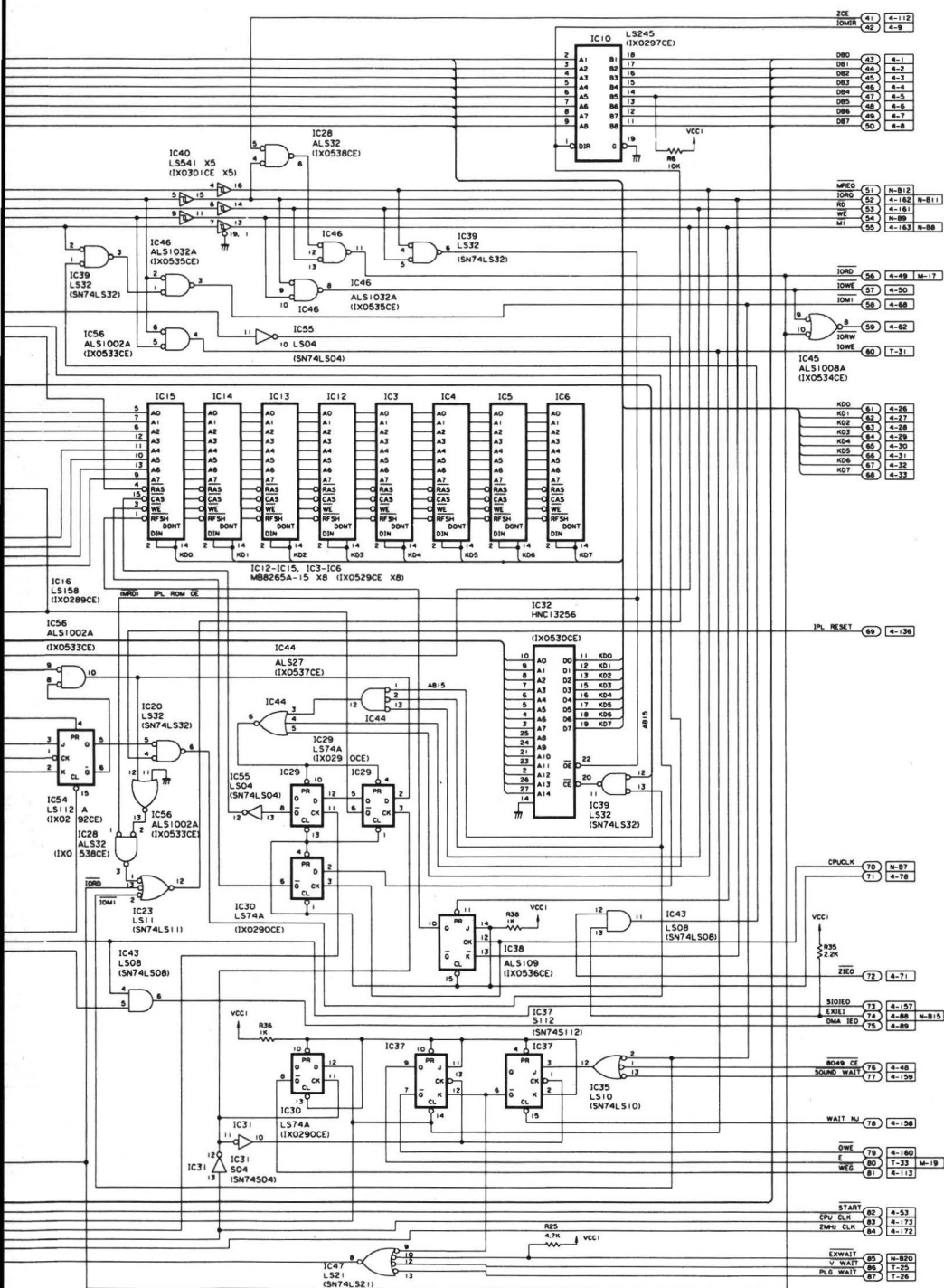
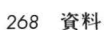
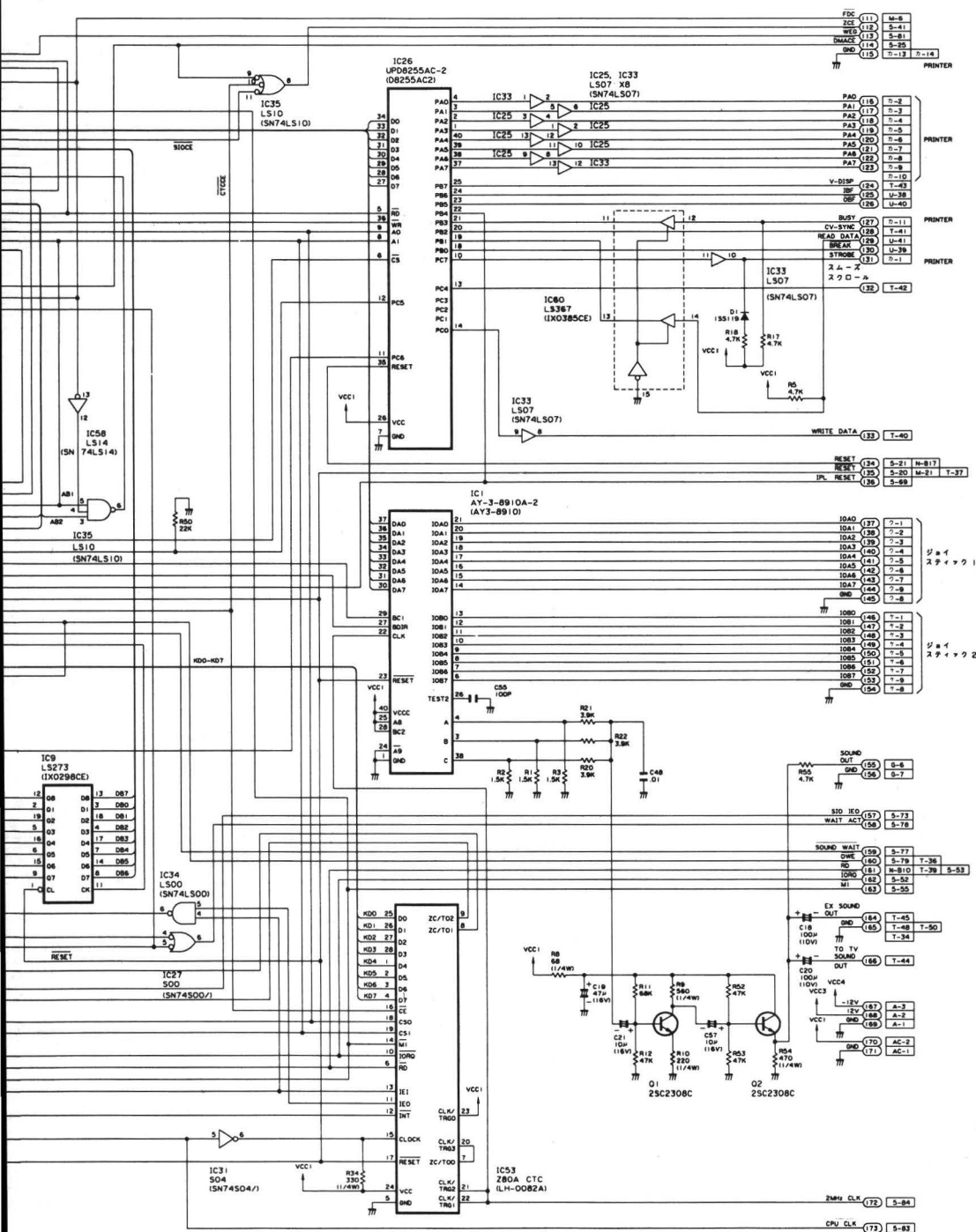
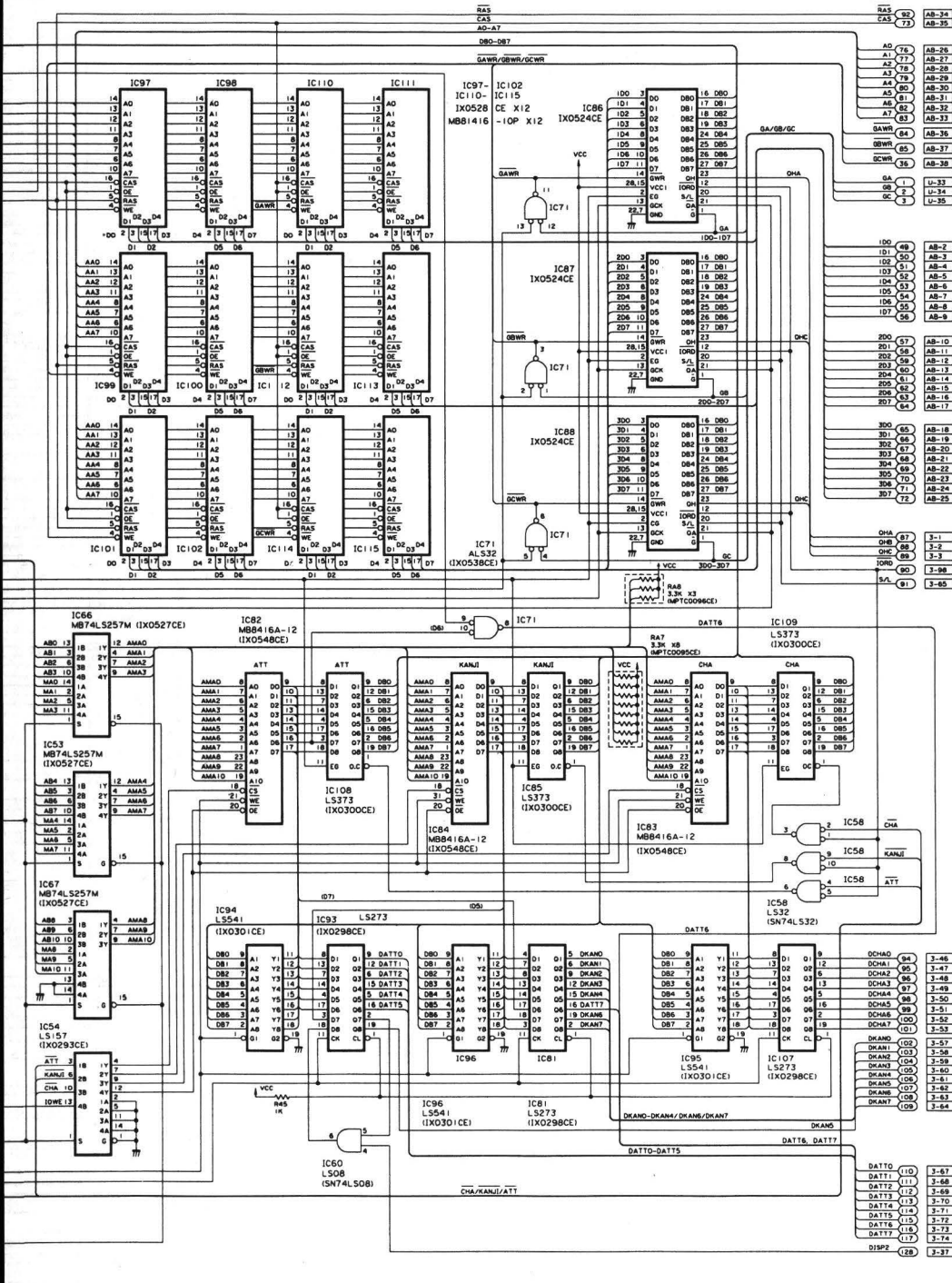
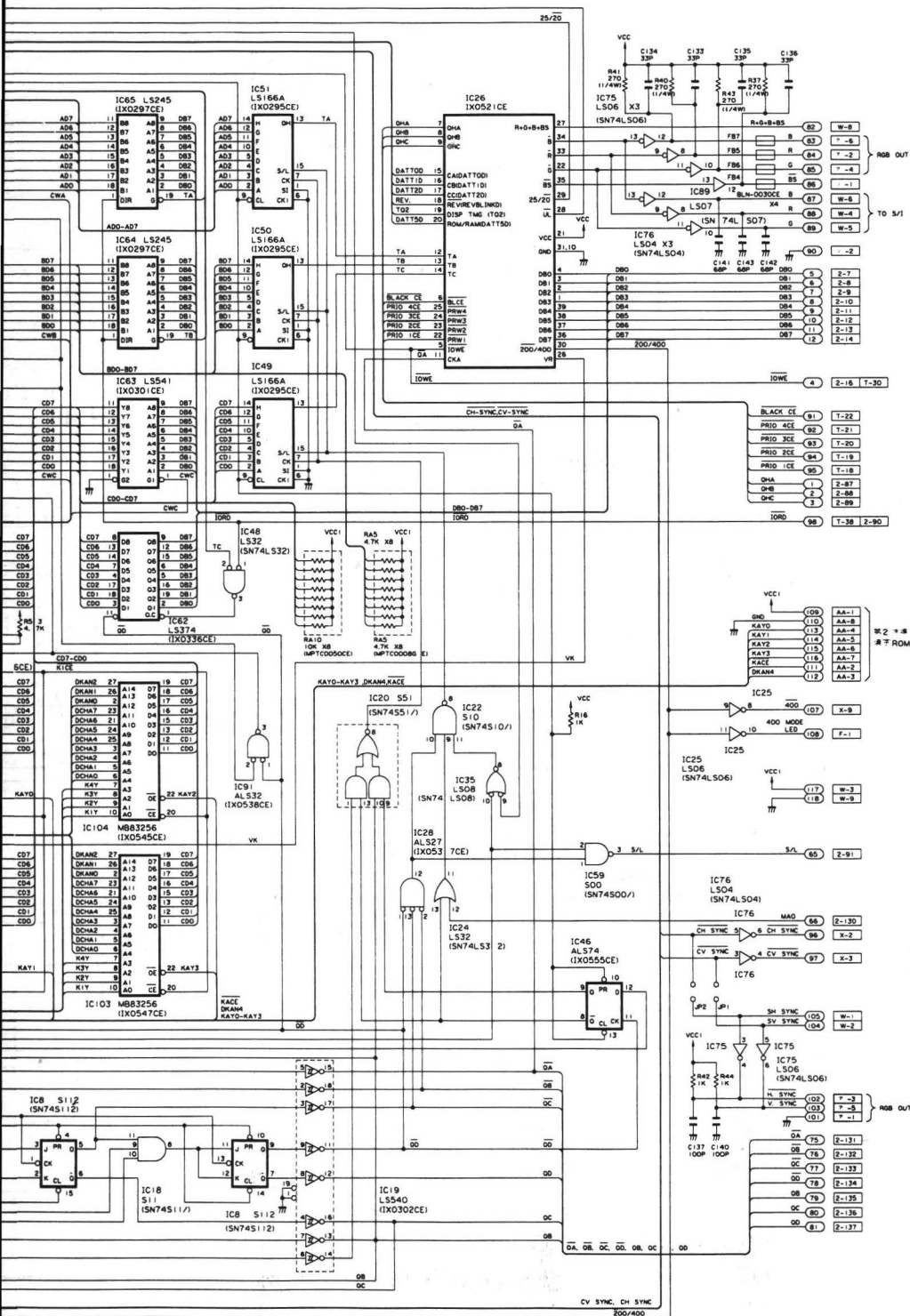
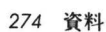
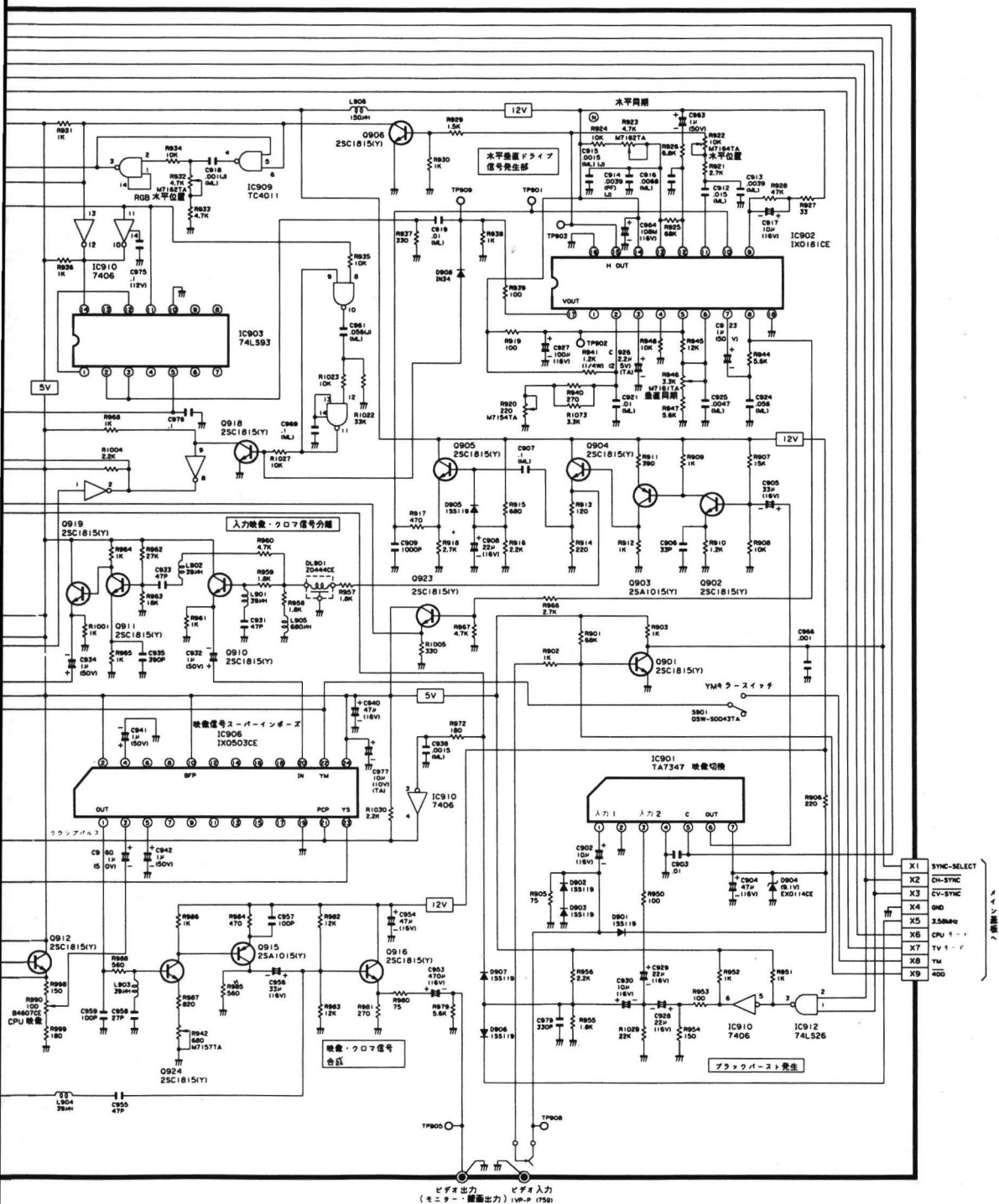
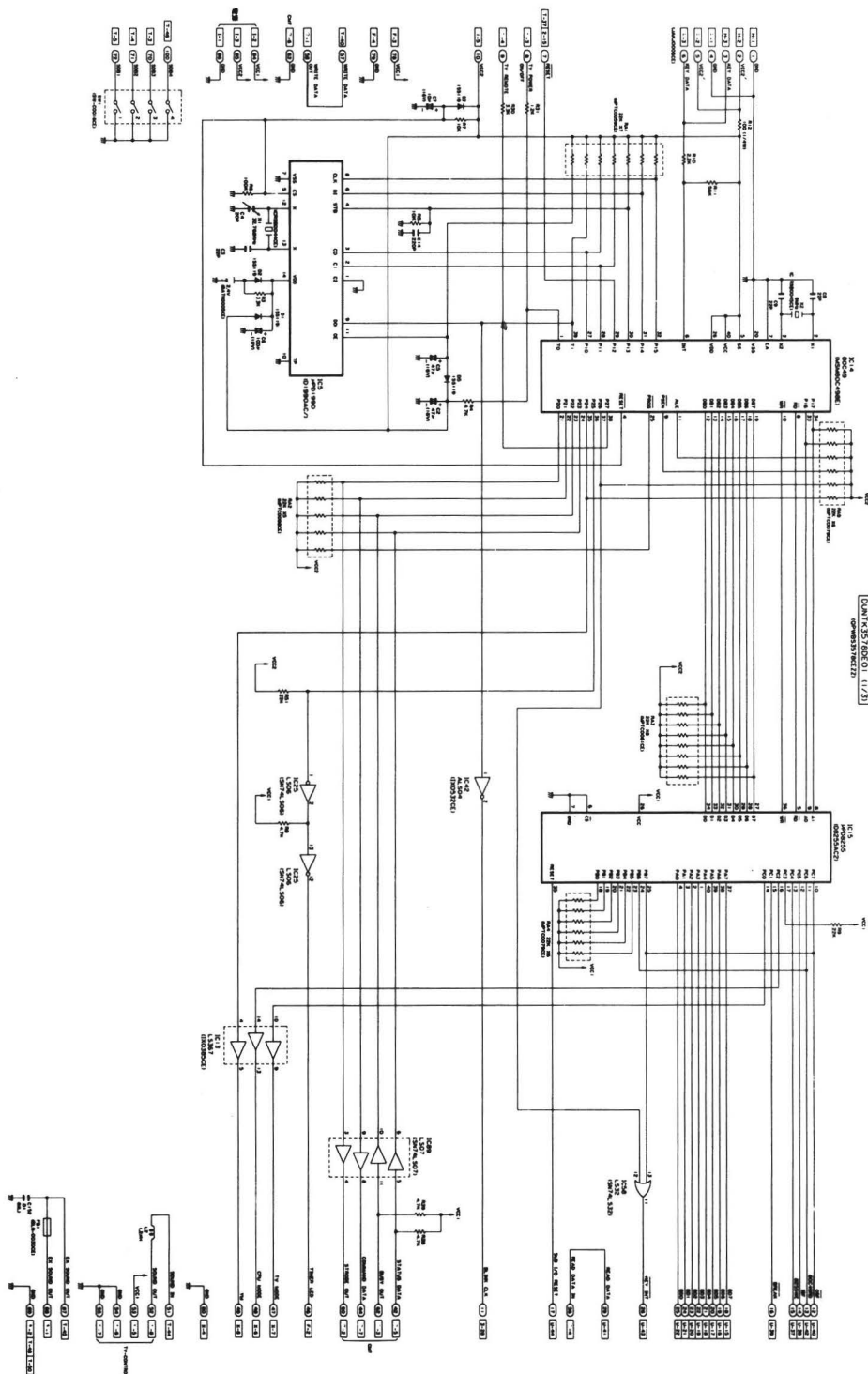
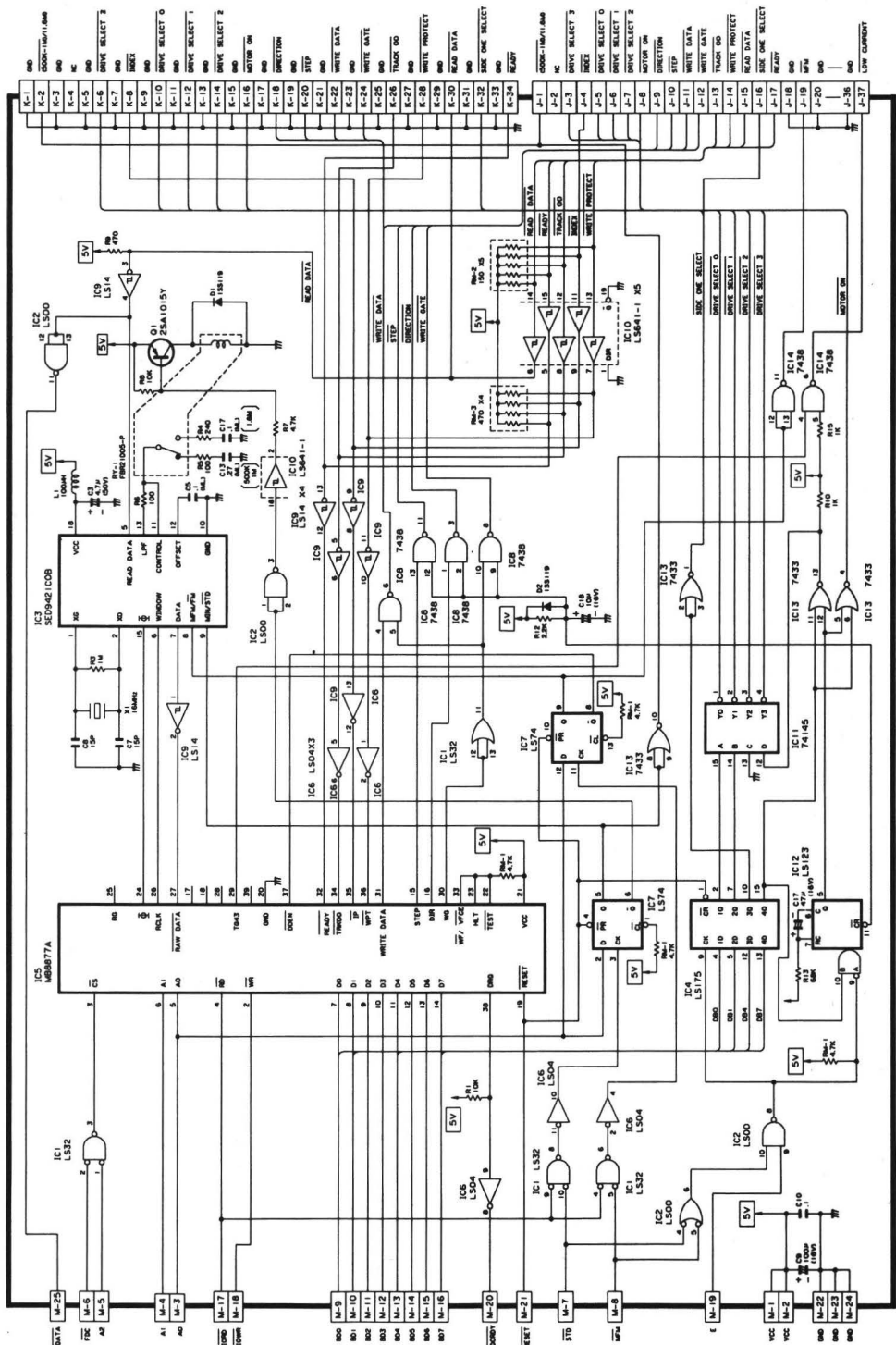
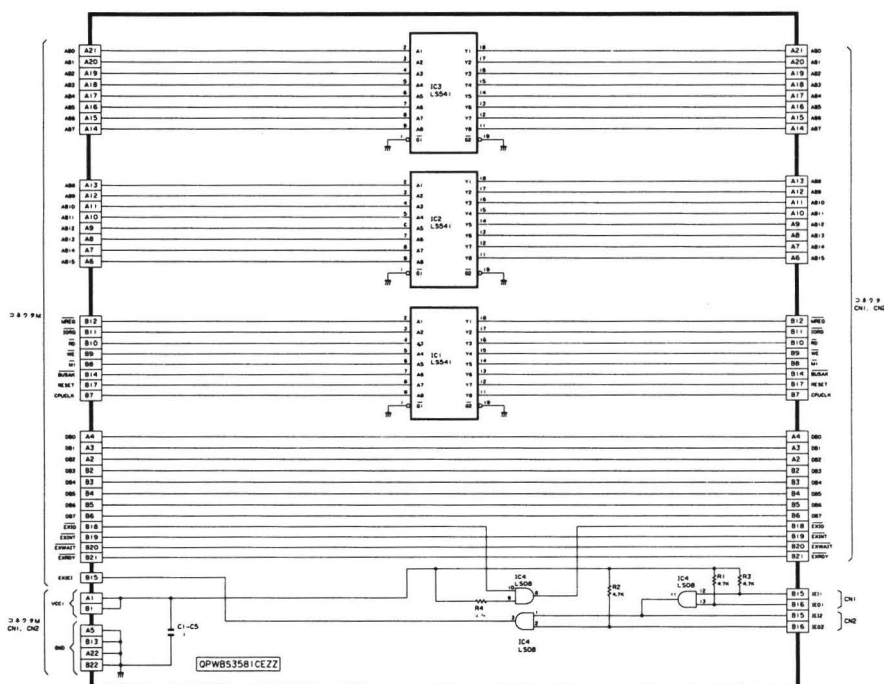
STR 7	STR 6	STR 5	STR 4	STR 3	STR 2	STR 1	STR 0

ステイタスの意味

コマンド	ステイタス	意味
タイプIコマンド	NOT－READY（STR 7）	NOT－READY＝1でディスクドライブが動作可能状態でない事を示します。これは，$\overline{READY}$とMRの論理和です
	WRITE－PROTECT（STR 6）	WRITE－PROTECT＝1でディスクへの書き込みが禁止されている事を示します。これは，$\overline{WPRT}$入力の反転コピーです
	HEAD－ENGAGED（STR 5）	HEAD－ENGAGED＝1でヘッドがメディアに押しつけられている事を示します。これはHLDとHLTの論理積です
	SEEK－ERROR（STR 4）	SEEK－ERROR＝1でベリファイ動作が成功しなかった事を示します。これは，IDフィールドが検出されなかったか，IDフィールドのトラック番号とトラックレジスタの内容が一致しない事によります
	CRC-ERROR（STR 3）	CRC－ERROR＝1で，IDフィールド読み出し時に読み出しエラーがあった事を示します
	TRACK 00（STR 2）	TRACK 00 ＝1でディスクのヘッドがトラック0の上にある事を示します。TRACK 0 0は$\overline{TR00}$入力の反転コピーです
	INDEX（STR 1）	INDEX＝1でインデックスホールを検出した事を示します。これはIP入力の反転コピーです
	BUSY（STR 0）	BUSY＝1でFDCがコマンド動作中である事を示します
タイプII／IIIコマンド	NOT-READY（STR 7）	NOT－READY＝1でディスクドライブが動作可能状態でない事を示します。これは$\overline{READY}$とMRの論理和です
	WRITE－PROTECT（STR 6）	WRITE－PROTECT＝1でディスクへの書き込みが禁止されている事を示します（$\overline{WPRT}$の反転コピー）
	RECORD TYPE /WRITE FAULT（STR 5）	リード動作の時，RECORD TYPEの表示として，DAMがDDMの時セットされます。ライト動作の時，WRITE FAULTの表示として，書き込み動作が打ち切られた事を示します。この時ステイタスは$\overline{WF}$の反転コピーでラッチされます
	RECORD NOT-FOUND（STR 4）	RECORD-NOT-FOUND＝1で指定されたトラック番号，サイド番号，セクタ番号を持ち，正しく読み出されたIDフィールドがなかった事を示します
	CRC-ERROR（STR 3）	CRC－ERRŌR＝1でディスクからIDフィールドもしくはデータフィールドの読み出しにおいて，読み出しエラーがあった事を示します
	LOST-DATA（STR 2）	LOST－DATA＝1で所要時間内にDRの読み出し，もしくは書き込みが行われなかったデータがあった事を示します
	DATA－REQUEST（STR 1）	DATA－REQUEST＝1て，FDCがプロセッサに対してDRの読み出し，もしくは書き込みを要求している事を示します。これはDRQのコピーです
	BUSY（STR 0）	BUSY＝1でFDCがコマンド実行中である事を示します

■X1turbo全回路図

●この回路図は，日本ソフトバンク出版部の責任において掲載します。回路図に関するメーカーへの，直接のお問い合わせはご遠慮下さい。

回路図① 内蔵カセット(モデル10のみ)

回路図 ⑩　オーディオアンプ CRTモード切り替えスイッチ

●この回路図は，日本ソフトバンク出版部の責任において掲載します。
回路図に関するメーカーへの，直接のお問い合わせはご遠慮下さい。

回路図① 漢字ROM部

セットインタフェース部

回路図⑤ I/Oスロット部

グラフィックV-RAM部

図 表 索 引

A

B

C

D

E

F

G

H

I

T

W

X

Y

Z

NUM